Spectrum™ Technology Platform

Version 9.0 SP3

API Guide

Pitney Bowes Software

Contents

Chapter 1: Getting Started....11

General Steps for Using the API....12

How Data is Passed to a Service....12

Supported Compilers....14

Third-Party Libraries....17

Network Protocols and Ports....17

Using the Sample Applications....17

Using SSL....18

Increasing the Timeout Value....18

Chapter 2: The C API....19

Introduction to the C API....20

Supported Libraries....20

Constants....24

Error Messages....26

Example Application....27

Server....30

CreateServer....30

DeleteServer....31

SetConnectionProperty....31

ServerConnect....32

ServerDisconnect....32

GetServiceFromServer....33

Service....34

ProcessMessage....34

Message....35

CreateMessage....35

DeleteMessage....35

GetContext....36

GetContextMap....36

PutContext....37

PutContextMap....38

SetContextMap....39

GetOption....40

GetOptions....41

PutOption....42
PutOptions....42
SetOptions....43
GetError....44
GetDataTable....45
DataTable....45
CreateDataTable....45
DeleteDataTable....46
AddColumn....46
GetColumnNames....47
GetColumnIndex....48
GetColumnCount....48
Clear....49
GetDataRows....49
AddRow....50
NewRow....50
GetRowCount....51
MergeDataTable....51
DataRow....52
CreateDataRow....52
DeleteDataRow....52
GetColumnNamesFromRow....52
GetColumnIndexFromRow....53
GetColumnCountFromRow....54
GetByIndex....54
GetByName....55
MergeDataRow....56
SetByName....56
SetByIndex....57
AddChild....58
GetChildren....59
ListChildNames....60
SetChildren....60

Chapter 3: The C++ API....63
Introduction to the C++ API....64
Supported Libraries....64
Constants....68
Error Messages....69
SmartPointer....70
Example Application....70
Server....72
Constructors....73
Destructor....73
Connect....73

Disconnect....73
SetConnectionProperty....74
GetService....74
Service....75
Process....75
Message....76
Constructors....76
Destructor....76
GetContext....77
GetContext....78
PutContext....78
PutContext....79
SetContext....79
GetOption....80
GetOptions....80
PutOption....81
PutOptions....82
SetOptions....82
GetError....83
GetDataTable....83
DataTable....84
Constructors....84
Destructor....84
AddColumn....85
GetColumnNames....85
GetColumnIndex....86
GetColumnCount....86
Clear....87
Iterator....87
AddRow....88
NewRow....88
GetRowCount....88
Merge....89
DataRow....89
Constructor....89
Destructor....90
GetColumnNames....90
GetColumnIndex....91
GetColumnCount....92
Get....92
Get....93
Merge....93
Set....94
Set....94
AddChild....95
GetChildren....95

ListChildNames....96
SetChildren....96

Chapter 4: The COM API....99
Introduction....100
Constants....100
Error Messages....101
Example Application....102
Server....103
Connect....103
Disconnect....103
GetService....104
SetConnectionProperty....104
Service....105
Process....105
Message....106
GetContext....106
GetContextMap....106
PutContext....107
PutContextMap....107
SetContextMap....108
GetOption....108
GetOptions....109
PutOption....109
PutOptions....110
SetOptions....110
GetError....110
GetDataTable....111
DataTable....111
AddColumn....111
GetColumnNames....112
GetColumnIndex....112
GetColumnCount....113
Clear....113
Iterator....113
AddRow....114
NewRow....115
GetRowCount....115
Merge....115
DataRow....116
GetColumnNames....116
GetColumnIndex....116
GetColumnCount....117
GetByIndex....117
GetByName....118

Merge....118
SetByName....118
SetByIndex....119
AddChild....120
GetChildren....120
ListChildNames....120
SetChildren....121
Map....122
Reset....122
Next....122
GetKey....123
GetValue....123

Chapter 5: The Java API....125
Introduction....126
Constants....126
Error Messages....127
Example Application....127
Server....128
Connect....128
Disconnect....132
SetConnectionProperty....132
GetService....133
Service....133
Process....133
Message....134
GetContext....134
GetContext....135
PutContext....135
PutContext....135
SetContext....136
GetOption....136
GetOptions....137
PutOption....137
PutOptions....137
SetOptions....138
GetError....138
GetDataTable....139
DataTable....139
AddColumn....139
GetColumnNames....139
GetColumnIndex....140
GetColumnCount....140
Clear....141
Iterator....141

AddRow....141
NewRow....142
GetRowCount....142
Merge....143
DataRow....143
GetColumnNames....143
GetColumnIndex....144
Get....144
Get....144
Merge....145
Set....145
AddChild....146
GetChildren....146
ListChildNames....147
SetChildren....147
Set....147

Chapter 6: ManagementAPI Methods....149
Introduction....150
GetLicenseInfo....150
GetVersionInfo....151

Chapter 7: The .NET API....153
Introduction....154
Constants....154
Error Messages....155
Example Application....155
Server....156
Connect....157
Disconnect....157
SetConnectionProperty....158
GetService....158
Service....159
Process....159
Message....159
GetContext....159
GetContext....160
PutContext....160
PutContext....160
SetContexts....161
GetOption....161
GetOptions....162
PutOption....162
PutOptions....162
SetOptions....163

GetError....163
GetDataTable....164
EnhancedDataTable....164
AddChild....164
GetChildren....165
ListChildNames....166
SetChildren....166

Chapter 8: Module Services....167
Address Now Module....168
Address Now Module....168
BuildGlobalAddress....168
GetGlobalCandidateAddresses....178
ValidateGlobalAddress....183
The ACR Code....202
Enterprise Geocoding Module....204
Enterprise Geocoding Module....204
Geocode Address Global....212
GeocodeAddressWorld....212
GeocodeUSAddress....242
GNAFPIDLocationSearch....288
ReverseAPNLookup....298
Reverse Geocode Address Global....309
ReverseGeocodeUSLocation....309
Geocode US Address Auxiliary Files....323
Location and Match Codes for U.S. Geocoding....328
Match Codes for U.S. Geocoding....345
Result Codes for International Geocoding....348
Encountering False Positives....351
Enterprise Tax Module....354
Enterprise Tax Module....354
AssignGeoTAXInfo....358
CalculateDistance....397
ReverseGeoTAXInfoLookup....399
Summary Reports....420
Match and Location Codes....425
Payroll Tax Correspondence File....434
User-Defined Data Files....437
GeoConfidence Module....443
GeoConfidence Module....443
GeoConfidenceSurface....444
Universal Addressing Module....446
Universal Addressing Module....446
AutoCompleteLoqate....450
GetCandidateAddresses....454

GetCandidateAddressesLoqate....461
GetCityStateProvince....465
GetCityStateProvinceLoqate....467
GetPostalCodes....469
GetPostalCodes Loqate....471
ValidateAddress....473
ValidateAddressAUS....528
ValidateAddressGlobal....533
ValidateAddressLoqate....549
Encountering False Positives....568
ValidateAddress Confidence Algorithm....570
Universal Name Module....574
OpenNameParser....574

Chapter 9: About Spectrum Technology Platform....581
What Is Spectrum™ Technology Platform?....582
Enterprise Data Management Architecture....583
Spectrum™ Technology Platform Architecture....586
Modules and Components....589

Appendix....593

Appendix A: Country ISO Codes and Module Support....595
Country ISO Codes and Module Support....596

1

Getting Started

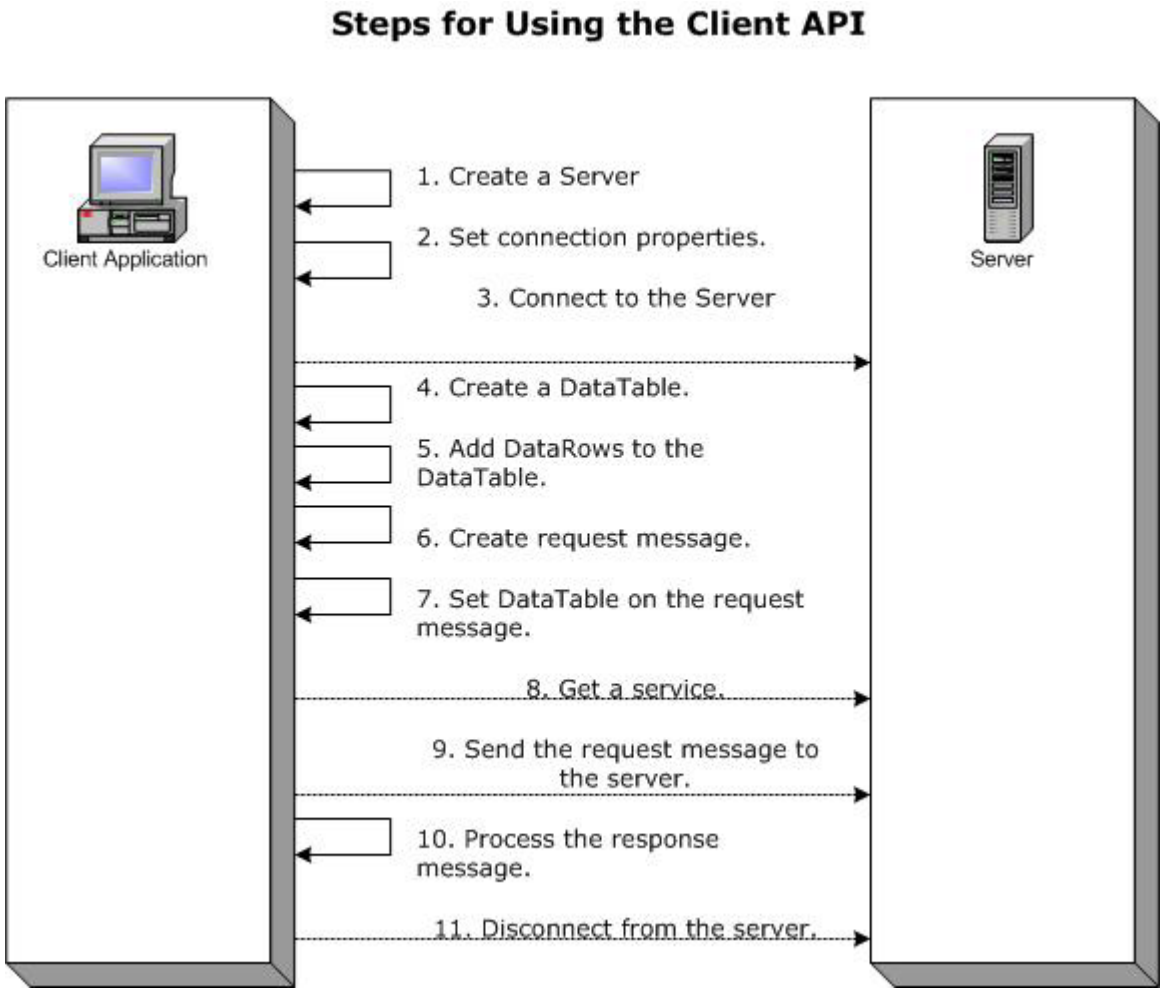
In this section:

- General Steps for Using the API 12
- How Data is Passed to a Service 12
- Supported Compilers 14
- Third-Party Libraries 17
- Network Protocols and Ports 17
- Using the Sample Applications 17
- Using SSL 18
- Increasing the Timeout Value 18

General Steps for Using the API

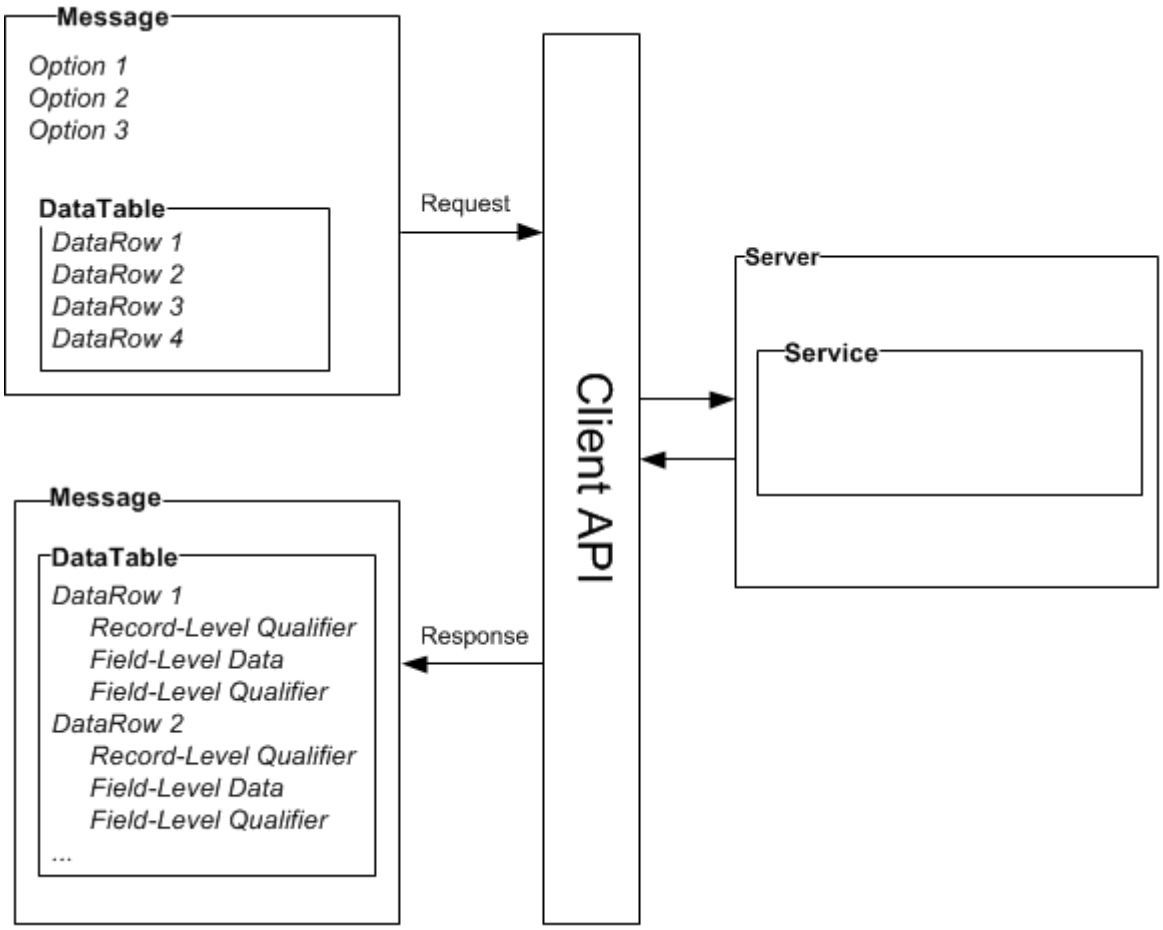
The basic steps for using the Spectrum™ Technology Platform API are:

1. Create a Server instance.
2. Set connection properties (connection type, host, port, etc.).
3. Connect to the Server.
4. Create a DataTable.
5. Add records to the DataTable.
6. Create a request message.
7. Set DataTable on the request message.
8. Get a service.
9. Send the request message to the Server.
10. Process the response message.
11. Disconnect from the Server.


Steps for Using the Client API
Client Application
Server
1. Create a Server
2. Set connection properties.
3. Connect to the Server
4. Create a DataTable.
5. Add DataRows to the DataTable.
6. Create request message.
7. Set DataTable on the request message.
8. Get a service.
9. Send the request message to the server.
10. Process the response message.
11. Disconnect from the server.

How Data is Passed to a Service

The following diagram illustrates how data is passed to a service through the API:



Message

Use the Message component to send your input data to the Spectrum™ Technology Platform service and receive output data from the service.

You can also use the Message component to override a service's default processing options. The default options for a service are set in Management Console. For example, the service ValidateAddress is capable of producing output in either mixed case or upper case. Let's say that in most instances, you need upper case output. However, one of your applications requires output in mixed case. In this situation, you would set the default for casing in the ValidateAddress service to upper case and override the default casing setting for that single application using the API. For those requests that should be handled according to the defaults you have specified, it is not necessary to provide any input options in your request.

The properties for Message include context properties, such as account ID, account password, service name, and service method; option properties, which are the service-specific runtime options; and error properties, which are the error class, error message, and error stacktrace.

DataTable

The DataTable component contains the records for your input and output data. Using the methods associated with this class, you define the column names for your output and add records to the data set. The Reset and Next methods are used to iterate over the results that are returned in a response from the server.

DataRow

The DataRow contains schema information and a list of data rows. Individual records reside in rows of data. For each output data row there are record-level qualifiers, field-level data, and field-level qualifiers.

Record-level qualifiers describe the processing of the record. Record-level qualifiers include status of the request (Success, Failure, or Error) and confidence in the accuracy of the output record.

Field-level data contains the validated, standardized, or enhanced record.

Field-level qualifiers include additional data about a given field. For example, the type of a Private Mailbox according to USPS categorization is a field-level qualifier.

Server

The Server component represents the Spectrum™ Technology Platform server. Using a Server component, you connect, disconnect, or access a specific service from the server.

Service

The Service component is used to process the message you're sending (i.e., send the input message and get back the response). The Service component has just one method: Process message.

Supported Compilers

The Spectrum™ Technology Platform Client SDK is supported with the following compiler and runtime minimum versions.

Java

Client SDK Package Directory: `clientSDK/platforms/java`

Client SDK requires the Java JDK, version 1.4 or higher. This is not installed with the Client SDK.

Windows

- JDK: 1.4
- C Compiler: MSVC 6.0 SP3, MSVC 2003, MSVC 2005
- C++ Compiler: MSVC 6.0 SP3, MSVC 2003, MSVC 2005
- C# .NET: Microsoft .NET Framework 1.1
- Visual Basic: MS Visual Basic 6.0

HP-UX RISC

- JDK: 1.4
- C Compiler: cc: HP92453-01 A.11.01.21 HP C (Bundled) Compiler
- C++ Compiler: aCC: HP aC++ B3910B A.03.30 HP aC++ B3910B A.03.27

The clientSDK 32 bit lib is linked to the following libraries:

- libpthread.1
- librt.2
- libnsl.1
- libxti.2

The clientSDK 64 bit lib is linked to the following libraries:

- libpthread.1

- libnsl.1
- librt.2
- libdl.1
- libc.2
- libxti.2
- libdl.1

HP-UX Itanium

- JDK: 1.4
- C Compiler: cc: HP aC++/ANSI C B3910B A.06.05
- C++ Compiler: aCC: HP aC++/ANSI C B3910B A.06.05

The clientSDK 32 bit lib is linked to the following libraries:

- libpthread.so.1
- libnsl.so.1
- librt.so.1
- libxti.so.1
- libdl.so.1

The clientSDK 64 bit lib is linked to the following libraries:

- libpthread.so.1
- libnsl.so.1
- librt.so.1
- libxti.so.1
- libdl.so.1

Red Hat (32 bit)

- Operating System: Red Hat Linux 2.4.9-e.65smp
- C Compiler: gcc version 2.96 (gcc 4.1 required for the Address Now Module)
- C++ Compiler: g++ version 2.96

The clientSDK lib is linked to the following libraries:

- libstdc++-libc6.2-2.so.3
- libm.so.6
- libc.so.6
- ld-linux.so.2

Red Hat (64 bit)

- Operating System: Red Hat Linux version 2.6.9-34.0.2.ELsmp
- C Compiler: gcc version 3.4.5
- C++ Compiler: g++ version 3.4.5

The clientSDK lib is linked to the following libraries:

- libstdc++.so.6
- libm.so.6
- libgcc_s.so.1
- libpthread.so.0
- libc.so.6
- ld-linux-x86-64.so.2

SuSE

- Operating System: SuSE SLES 8 (powered by UnitedLinux 1.0) (i586)\nKernel 2.4.21-295-smp (0).
- C Compiler: gcc version 3.2.2
- C++ Compiler: g++ version 3.2.2

The clientSDK lib (32 bit) is linked to the following libraries:

- libstdc++.so.5
- libm.so.6
- libgcc_s.so.1
- libc.so.6
- ld-linux.so.2

Solaris

- Operating System: Solaris 5.8
- C Compiler: cc: Forte Developer 7 C 5.4 2002/03/09
- C++ Compiler: CC: Forte Developer 7 C++ 5.4 Patch 111715-16 2005/04/28

The clientSDK 32 bit lib is linked to the following libraries:

- libpthread.so.1
- libsocket.so.1
- libnsl.so.1
- librt.so.1
- libc.so.1
- libdl.so.1
- libmp.so.2
- libaio.so.1
- libc_psr.so.1

The clientSDK 64 bit lib is linked to the following libraries:

- libpthread.so.1
- libsocket.so.1
- libnsl.so.1
- librt.so.1
- libc.so.1
- libmp.so.2
- libmd5.so.1
- libscf.so.1
- libaio.so.1
- libdoor.so.1
- libuutil.so.1
- libm.so.2
- libc_psr.so.1
- libmd5_psr.so.1

AIX

- Operating System: AIX version 5.1.0.0
- C Compiler: xlc 6.0 Visual Age C 6.0
- C++ Compiler: xlC 6.0 Visual Age C++ 6.0

The clientSDK 32 bit and 64 bit lib are linked to the following libraries:

- libC.a
- libc_r.a
- libpthread.a
- librtl.a

Third-Party Libraries

The Spectrum™ Technology Platform API uses the following third-party libraries.

- Apache Commons Pool 1.6
- ICU 3.2.0
- Jakarta Commons HttpClient 3.1
- OpenSSL 1.0.1e
- OpenTop 1.5.3
- POCO 1.3

Network Protocols and Ports

The API communicates with the Spectrum™ Technology Platform server using HTTP, HTTPS, or SOCKET. Spectrum™ Technology Platform typically uses port 8080 to listen for HTTP requests and port 443 for HTTPS requests. Secure Socket Layer (SSL) and Proxy (HTTP, HTTPS) features are also supported in the C, C++, COM, Java, and .NET APIs. .NET, Java, and COM APIs support Unicode; C and C++ APIs support both ASCII and Unicode.

In addition to HTTP, Spectrum™ Technology Platform supports a persistent SOCKET connection. The high-speed SOCKET connection provides much faster performance than traditional HTTP. Spectrum™ Technology Platform typically uses port 10119 to listen for SOCKET requests.

Using the Sample Applications

The Client SDK includes sample applications for all supported languages. The sample applications call a sample service on the Spectrum™ Technology Platform server which changes the casing of the input data to either upper case or lower case.

1. Copy the `casing-<version>.car` file from `ClientAPI\common\lib` to the `server\app\deploy` folder on the Spectrum™ Technology Platform server.

   The casing service used by the sample applications is now deployed on your Spectrum™ Technology Platform server.

2. In the `ClientAPI\platforms` folder, find the `samples` subfolder for your platform and open the `readme.txt` file for further instructions on using the sample applications.

Note: You can modify the sample application to use one of the services you have licensed, and recompile the sample to run.

Using SSL

This procedure describes how to use SSL communication between your application and the Spectrum™ Technology Platform server.

1. Specify the root CA that will be used for communication between your application and the Spectrum™ Technology Platform server by doing one of the following:

Option	Description
If you do not know which root CA will be used	Copy the file `ca-bundle.pem` to your working directory. For C/C++ and COM, and ASP, the .pem file is located in the following folder in the location where you installed the Client SDK: `Spectrum Client SDK\ClientAPI\platforms\windows\c-c++\<32or64>\<version>\lib\openssl` For ASP, some examples of a working folder are: • If you use Internet Information Services to run ASP, copy `ca-bundle.pem` to the Windows system directory (for example, `C:\Windows\system32`). • If you use Internet Explorer to run ASP, copy `ca-bundle.pem` to the Internet Explorer default working directory (for example, `C:\Documents and Settings\<user>\Desktop`).
If you know which root CA will be used	Specify the root CA certificate in your CA bundle file.

2. In your application, when you connect to the server set the connection type to HTTPS.

Increasing the Timeout Value

If you experience timeouts between the client and server, you can increase the timeout value for the client.

- Use the `setConnectionProperty` method to set the timeout value.

2

The C API

In this section:

- Introduction to the C API 20
- Server 30
- Service 34
- Message 35
- DataTable 45
- DataRow 52

Introduction to the C API

The C API consists of the following structures:

- Server
- Service
- Message
- DataTable
- DataRow

Note: The C API is a C wrapper around the C++ code. On Unix you can use a C++ compiler to build your C application, which is the preferred approach. However, a C compiler can also be used directly on Linux, and Solaris. On HP-UX and AIX, you need to link all the C++ required libs when you use the C compiler. To do this, run `ldd ./batch` under `..../samples/batch/bin/` to get the list of all dependent libs and put them in the link section of your makefile.

Supported Libraries

Spectrum™ Technology Platform provides an ASCII and Unicode version C API, while the Unicode version remains as compatible as possible with the original ASCII-version API design. Spectrum™ Technology Platform applies International Components for Unicode (ICU) in the API to support the Unicode feature. ICU is a mature, widely used set of C/C++ libraries for Unicode support and is developed by IBM.

The Unicode standard defines a default encoding based on 16-bit code units. This is supported in ICU by the definition of the UChar to be an unsigned 16-bit integer type (unsigned short *). This is the base type for character arrays for strings in ICU. Spectrum™ Technology Platform uses UChar as the Unicode string representation in our C API.

Note: Not all services support the full Unicode character set. For example, the ValidateAddress service supports the ISO 8859-1 character set for US input and international input and output, and the CP 850 character set for Canadian input and output. However, the Unicode libraries should be used whenever your input data may contain any non-ASCII character, even if the underlying service does not support the full Unicode character set.

For detailed information about UChar, please refer to the following two sites:

- **icu.sourceforge.net/userguide**
- **www.ibm.com/software/globalization/icu**

C Libraries Supported on Windows

Each API configuration produces library files with a common base name (g1client) but with a unique suffix and possibly prefix ("lib" in the case of static libraries). The library suffixes work like this:

```
<lib>g1client<S><U><D>.<lib|dll>
```

- lib—indicates a static library.
- dll—indicates a dynamic (shared) library.
- S—indicates a single-threaded build. If this is absent it indicates a multi-threaded build.
- U—indicates a UNICODE version build. If this is absent it indicates an ASCII build.
- D—indicates a debug build. If this suffix is absent it indicates an optimized release build.

To enable the UNICODE version, the LIB_UNICODE macro definition must be in your project.

To use the static C/C++ API library UNICODE version, you need to define U_STATIC_IMPLEMENTATION in your project.

To use the dynamic version, you need to define G1CLIENT_DLL in your project.

We also provide a file called "auto_link.h" in the header file directory and it automatically links to all the corresponding libraries according to the project settings.

To call 64-bit libraries in Windows, you need to define VER_64 in your project.

Static Library

Note: The names provided in this section are for 32-bit libraries. For 64-bit libraries, replace "32" with "64" in the library name.

Table 1: Single Threaded/Release

	ASCII	Unicode
g1	libg1client_S.lib	libg1client_SU.lib
openssl	otlibeay32.lib otlibssl32.lib	otlibeay32.lib otlibssl32.lib
opentop	opentop.lib	opentopw.lib
icu		libicuuc.lib libicudt.lib libicuin.lib libicuio.lib
Poco	PocoXML32.lib	PocoXML32w.lib

Table 2: Single Threaded/Debug

	ASCII	Unicode
g1	libg1client_SD.lib	libg1client_SUD.lib
openssl	otlibeay32d.lib otlibssl32d.lib	otlibeay32d.lib otlibssl32d.lib
opentop	opentopd.lib	opentopwd.lib
icu		libicuucd.lib libicudtd.lib libicuind.lib libicuiod.lib
Poco	PocoXML32d.lib	PocoXML32wd.lib

Table 3: Multi/Release (using Multi-Threaded CRT)

	ASCII	Unicode
g1	libg1client.lib	libg1client_U.lib
openssl	otlibeay32mt.lib otlibssl32mt.lib	otlibeay32mt.lib otlibssl32mt.lib
opentop	opentopmt.lib	opentopmtw.lib
icu		libicuucmt.lib libicudtmt.lib libicuinmt.lib libicuiomt.lib
Poco	PocoXMLmt32.lib	PocoXML32mtw.lib

Table 4: Multi/Debug (using Multi-Threaded CRT)

	ASCII	Unicode
g1	libg1client_D.lib	libg1client_UD.lib
openssl	otlibeay32mtd.lib otlibssl32mtd.lib	otlibeay32mtd.lib otlibssl32mtd.lib
opentop	opentopmtd.lib	opentopmtwd.lib
icu		libicuucmtd.lib libicudtmtd.lib libicuinmtd.lib libicuiomtd.lib
Poco	PocoXMLmt32d.lib	PocoXML32mtwd.lib

Dynamic Library

Note: The names provided in this section are for 32-bit libraries. For 64-bit libraries, replace "32" with "64" in the library name.

Table 5: Multi/Release (using Multi-Threaded CRT)

	ASCII	Unicode
g1	g1client.dll	g1client_U.dll
openssl	otlibeay32mts.dll otlibssl32mts.dll	otlibeay32mts.dll otlibssl32mts.dll
opentop	opentopmts.dll	opentopmtws.dll
icu		icuuc32.dll icuio32.dll icuin32.dll icudt32.dll
Poco	PocoXML32mts.dll	PocoXML32mtws.dll

Table 6: Multi/Debug (using Multi-Threaded CRT)

	ASCII	Unicode
g1	g1client_D.dll	g1client_UD.dll
openssl	otlibeay32mtds.dll otlibssl32mtds.dll	otlibeay32mtds.dll otlibssl32mtds.dll
opentop	opentopmtds.dll	opentopmtwds.dll
icu		icuuc32d.dll icuio32d.dll icuin32d.dll icudt32d.dll
Poco	PocoXML32mtds.dll	PocoXML32mtwds.dll

C Libraries Supported on Unix

Each ClientSDK configuration produces library files with a common base name (libg1client) but with a unique suffix. Spectrum™ Technology Platform provides a multithread and release build for ASCII version and UNICODE version.

The library suffixes work like this:-

```
libg1client<U>.<so|sl|a>
```

Where U indicates a UNICODE version build. If this is absent it indicates an ASCII build.

To use the UNICODE version, you need to define LIB_UNICODE in your project.

In UNICODE Version C++ API, the namespace for all classes is g1client.

Table 7: AIX

	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libssl.so	libcrypto.so libssl.so
opentop	libopentop-xlCmt.so	libopentop-xlCmtw.so libotxml-xlCmtw.so
icu		libicudata34.a libicui18n34.a libicuio34.a libicuuc34.a
Poco	libPocoXML.so	

Table 8: HP-UX RISC

	ASCII	Unicode
g1	libg1client.sl	libg1client_U.sl
openssl	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl
icu		libicudata.sl libicudata.sl.34 libicui18n.sl libicui18n.sl.34 libicuio.sl libicuio.sl.34 libicuuc.sl libicuuc.sl.34
Poco	libPocoXML.sl	

Table 9: HP-UX Itanium

	ASCII	Unicode
g1	libg1client.sl	libg1client_U.sl
openssl	libcrypto.a libssl.a	libcrypto.a libssl.a
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl

	ASCII	Unicode
icu		libicudata.sl libicudata.sl.34 libicudata.sl.34.0 libicui18n.sl libicui18n.sl.34 libicui18n.sl.34.0 libicuio.sl libicuio.sl.34 libicuio.sl.34.0 libicuuc.sl libicuuc.sl.34 libicuuc.sl.34.0
Poco	libPocoXML.sl	

Table 10: Linux

	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-gccmt.so	libopentop-gccmtw.so libotxml-gccmtw.so
icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34 libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34
Poco	libPocoXML.so	

Table 11: Solaris SPARC

	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-fortemt.so	libopentop-fortemtw.so libotxml-fortemtw.so
icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34 libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34
Poco	libPocoXML.so	

Constants

The C API uses two sets of constants. The first set is for the Server component, described in the table below.

Table 12: Constants for the Server Component

Constant Name	Description/Default	Example
SERVER_HOST	String for server host name. Default is "localhost".	65.89.200.89
SERVER_PORT	String for server port. Default is "8080".	10119
SERVER_ACCOUNT_ID	String for server account ID. No default value.	user1
SERVER_ACCOUNT_PASSWORD	String for server account password. No default value.	user1
SERVER_CONNECTION_TIMEOUT	String for server connection timeout, in milliseconds. Default is "5000".	50000
SERVER_CONNECTION_TYPE	String for server connection type. Currently only supports HTTP, HTTPS, or SOCKET. Default is "HTTP".	HTTP(S)
SERVER_PROXY_HOST	String for proxy server host name. No default value.	192.168.1.77
SERVER_PROXY_PORT	String for proxy server port. No default value.	8080
SERVER_PROXY_USER	String for proxy server account ID. No default value.	user1
SERVER_PROXY_PASSWORD	String for proxy server account password. No default value.	user1

The second set of constants is for the Message component.

Table 13: Constants for the Message Component

Constant Name	Description	Example
MESSAGE_CONTEXT_ACCOUNT_ID	String for message context account ID.	user1
MESSAGE_CONTEXT_ACCOUNT_PASSWORD	String for message context account password.	user1
MESSAGE_CONTEXT_SERVICE_NAME	String for message context service name.	echoservice

Error Messages

Some functions return a SUCCESSFUL_RETURN or 0 (zero) value if they are successful. If it is not successful, the function returns an error code. In order to retrieve error messages, call getErrorMessage(int errorCode). For example:

```
Server *server = NULL;
int nRet;
//Create Server
server = createServer();
//set the property to the server
...
//Connect to server
printf("Making connection to the server...\n");
nRet = serverConnect(server);
if(nRet != SUCCESSFUL_RETURN)
{
// ASCII Version-use the following code
printf(getErrorMessage(nRet));
//Unicode Version -use the following code
UChar * error = getErrorMessage(nRet));
// more code to print out the error messageÂ….
 return ;
 }
```

The C API uses the following error messages.

- Error messages for passing a null structure:
  - "Input null DataRow"
  - "Input null DataTable"
  - "Input null Message"
  - "Input null Server"
- Error messages for connections:
  - "Connection type not supported"
  - "Client timeout"
  - "Blank connection property name"
  - "Blank property name"
- Error messages for creating a DataTable:
  - "Blank column name"
  - "Duplicated column name"
- Error messages for MessagePackaging Exception:
  - "Input Message is null"
  - "Failed to connect to Server"
  - "Failed to disconnect from Server"
  - "Failed to open Http Connection"
  - "Failed to get Service"
  - "Failed to package the message using Serializer and Encoding"

Example Application

The sample code shown below illustrates how to use the ASCII version C API.

```
// Declarations
Server *server = NULL;
Message *request = NULL;
DataTable *dataTable = NULL;
Message *reply = NULL;
Service *service = NULL;
int nRet;
DataRow *row1 = NULL;
DataRow *row2 = NULL;
DataTable *returnDataTable= NULL;
char** columnNames;
DataRow** rows;
DataRow*dataRow;
int i;
int j;
char* value;

//Create Server
server = createServer();

//Set server connection properties
nRet = setConnectionProperty(server, SERVER_HOST, "localhost");
nRet = setConnectionProperty(server, SERVER_PORT, "10119 ");
nRet = setConnectionProperty(server, SERVER_CONNECTION_TYPE,"SOCKET");
nRet = setConnectionProperty(server, SERVER_ACCOUNT_ID,"guest");
nRet = setConnectionProperty(server, SERVER_ACCOUNT_PASSWORD,"");

//Connect to server
nRet = serverConnect(server);
if(nRet != SUCCESSFUL_RETURN)
{
printf( getErrorMessage(nRet));
// free memory
if(server)
nRet = deleteServer(server);
return ;
}

 //Get Service From Server
 service = getServiceFromServer(server,"ValidateAddress" );

//Create Input Message
request = createMessage();

//Fill DataTable in the input message
dataTable = getDataTable(request);
nRet= addColumn( dataTable, "AddressLine1", &nRet);
nRet= addColumn( dataTable, "City", &nRet);
nRet= addColumn( dataTable, "StateProvince", &nRet);

row1 = newRow( dataTable );
setByIndex (row1, 0 , "4200 Parliament Place");
setByIndex (row1, 1 ,"Lanham");
setByIndex (row1, 2 , "Maryland");

addRow( dataTable, row1);

row2 = newRow( dataTable );
setByIndex (row2, 0 , "10535 Boyer Blvd");
setByIndex (row2, 1 ,"Austin");
setByIndex (row2, 2 , "Texas");

addRow( dataTable, row2);

//Set"option" Properties to the Input Message
```

```
nRet = putOption(request, "OutputCasing","M");
nRet = putOption(request, "OutputRecordType","A");

//Process Input Message, return output Message
nRet = processMessage(service, request, &reply);
if(nRet != SUCCESSFUL_RETURN)
{
printf("Error Occurred, " );
printf(getErrorMessage(nRet));

// free memory
if(request)
nRet = deleteMessage(request);
if(reply)
nRet = deleteMessage(reply);
if(server)
nRet = deleteServer(server);

return ;

}

//Disconnect from server
nRet = serverDisconnect(server);

//Get the result from the response message
returnDataTable = getDataTable(reply );
columnNames = getColumnNames(returnDataTable);

rows = getDataRows( returnDataTable);

for( i=0; i < getRowCount( returnDataTable); i++)
{
dataRow = rows[i];

for(j=0; j < getColumnCount(returnDataTable); j++)
{
value = (char*)getByIndex( dataRow, j);
printf(value);
printf("\n");
}
}

//Free Memory
if(request)
nRet = deleteMessage(request);
if(reply)
nRet = deleteMessage(reply);
if(server)
nRet = deleteServer(server);
}
```

The sample code shown below illustrates how to use the Unicode version C API. The string here is represented by UChar*(or unsigned short*), which is 16-bit type to represent the Unicode string .ICU provides a function called u_charsToUChars, which converts 8-bit string to 16-bit string. The example here shows how to call Unicode version C API. The input string are all ASCII, so that we use u_charsToUChars to convert to 16-bit string. You could also construct Unicode string to directly pass in C API.

```
UChar* convertcharToUChar( char* name, UChar* value)
  {
   int lenName= strlen(name);
   u_charsToUChars(name, value, lenName );
   value[ lenName]=0;
   return value;
  }

   // Declarations
   Server *server = NULL;
```

```
   Message *request = NULL;
   DataTable *dataTable = NULL;
   DataTable *returnDataTable= NULL;
   Message *reply = NULL;
   Service *service = NULL;
   int nRet;
   DataRow* newDataRow;
   UChar    name[128];
   UChar value[128];
   UChar** columnNames;
   DataRow** rows;
   DataRow* dataRow;
   int i, j;
   UChar* columnValue;
   UChar*   errorMsg;

   //Create Server
   server = createServer();

   //Set server connection properties
   setConnectionProperty(server, convertcharToUChar( SERVER_HOST, name) ,
 convertcharToUChar( "localhost", value));
   setConnectionProperty(server, convertcharToUChar( SERVER_PORT, name) ,
 convertcharToUChar( "10119", value));
  setConnectionProperty(server, convertcharToUChar( SERVER_CONNECTION_TYPE,
 name) , convertcharToUChar( "SOCKET", value));
   setConnectionProperty(server, convertcharToUChar( SERVER_ACCOUNT_ID,
name) , convertcharToUChar( "guest", value));
   setConnectionProperty(server, convertcharToUChar(
SERVER_ACCOUNT_PASSWORD, name) , convertcharToUChar( "", value));


   //Connect to server
   nRet = serverConnect(server);
   if(nRet != SUCCESSFUL_RETURN)
   {
   // error handling
   errorMsg = getErrorMessage(nRet);
   // free memory
   if(server)
   nRet = deleteServer(server);
   return ;
   }

   //Get Service From Server
   service = getServiceFromServer(server,convertcharToUChar(
"ValidateAddress", name));

   //Create Input Message
   request = createMessage();

   //Fill DataTable in the input message
   dataTable = getDataTable(request);
   addColumn( dataTable,convertcharToUChar( "AddressLine1", name),  &nRet);

   addColumn( dataTable,convertcharToUChar( "City", name),  &nRet);
   addColumn( dataTable,convertcharToUChar( "PostalCode", name),  &nRet);

   addColumn( dataTable,convertcharToUChar( "StateProvince", name),  &nRet);


   newDataRow = newRow( dataTable );

  setByIndex (newDataRow, 0 , convertcharToUChar( "74, Rue Octave Bénard",
 name) );
  setByIndex (newDataRow, 1 , convertcharToUChar( "Etang-Salé-les- Bains",
 name) );
   setByIndex (newDataRow, 2 , convertcharToUChar( "97427", name) );
  setByIndex (newDataRow, 3 , convertcharToUChar( "Reunion Island", name)
 );
```

```
   addRow( dataTable, newDataRow);

   //Set"option" Properties to the Input Message
   nRet = putOption(request, convertcharToUChar( "OutputCasing", name),
convertcharToUChar( "M", value));
   nRet = putOption(request, convertcharToUChar( "OutputRecordType", name),
 convertcharToUChar( "A", value));


   //Process Input Message, return output Message
   nRet = processMessage(service, request, &reply);
    if(nRet != SUCCESSFUL_RETURN)
   {
   // error handling
   errorMsg = getErrorMessage(nRet);
   // free memory
   if(request)
   nRet = deleteMessage(request);
   if(reply)
   nRet = deleteMessage(reply);
   if(server)
   nRet = deleteServer(server);

   return ;
   }

   //Disconnect from server
   nRet = serverDisconnect(server);

   //Get the result from the response message
   returnDataTable = getDataTable(reply );
   columnNames = getColumnNames(returnDataTable);
   rows = getDataRows( dataTable);
   for( i=0; i < getRowCount( dataTable); i++)
   {
   dataRow =  rows[i];
   for(j=0; j < getColumnCount(dataTable); j++)
   {
   columnValue = (UChar*)getByIndex( dataRow, j);
   }
   }

   //Free Memory
   if(request)
   nRet = deleteMessage(request);
   if(reply)
   nRet = deleteMessage(reply);
   if(server)
   nRet = deleteServer(server);
```

Server

The Server structure is used to connect to server, disconnect from the server, and get the service from the server.

CreateServer

Creates the server.

Syntax

```
Server* createServer()
```

Parameters

None.

Result

The server is created.

Example

```
Server *server = NULL;
//Create Server
server = createServer();
```

DeleteServer

Deletes the server.

Syntax

```
int deleteServer(Server* server)
```

Parameters

- Server— the server to be deleted.

Result

Returns 0 (if successful) or error code.

Example

```
int nRet;
nRet = deleteServer(server);
```

SetConnectionProperty

Establishes the server connection configuration properties, such as host name and length of timeout.

Syntax

ASCII Version

```
int setConnectionProperty(Server* server, const char* name, const char*
value)
```

Unicode Version

```
int setConnectionProperty(Server* server, const UChar* name, const UChar*
 value)
```

Parameters

- Server — the server to which the client connects
- Name — the name of the connection property, such as HOST
- Value — the value for the name of the connection property, such as "www.myhost.com"

Result

Returns 0 (if successful) or error code.

Example

ASCII Version

```
int nRet;
Server *server = NULL;
nRet = createServer(&server);
nRet = setConnectionProperty(server, SERVER_HOST,
"localhost");
```

Unicode Version

```
int nRet;
// construct 16-bit string
UChar serverHost[32];
char* SERVER_HOST= SERVER_HOST;
u_charsToUChars(SERVER_HOST, serverHost, strlen(SERVER_HOST));
serverHost [ strlen(SERVER_HOST)]=0;
// construct 16-bit string
UChar hostValue [32];
char* value= "localhost";
u_charsToUChars(value, hostValue, strlen(value));
hostValue[ strlen(value)]=0;
nRet = setConnectionProperty(server, serverHost , hostValue);
```

ServerConnect

Reads the properties to determine the configuration settings and makes a connection to the server.

Note: C uses the HTTP, HTTPS, or SOCKET server connection protocol. HTTP and HTTPS logically establish a client connection but do not actually connect to the server until a GetService or Process method is invoked. The SOCKET protocol establishes a connection to the server when Connect is invoked.

Syntax

```
int serverConnect(Server* server)
```

Parameters

- Server—the server to which the client connects

Results

Returns 0 (if successful) or error code.

Example

```
int nRet;
nRet = serverConnect(server);
```

ServerDisconnect

Disconnects from the server.

Syntax

```
int serverDisconnect(Server* server)
```

Parameters

- Server—the server from which the client disconnects.

Results

Returns 0 (if successful) or error code.

Example

```
int nRet;
nRet = serverDisconnect(server);
```

GetServiceFromServer

Gets the service from the server.

Note: See the Component Reference section of this guide for a list of services that may be available to you.

Syntax

ASCII Version

```
Service* getServiceFromServer(Server* server,const char* serviceName )
```

Unicode Version

```
Service* getServiceFromServer(Server* server,const UChar* serviceName )
```

Parameters

- Server - server from which the client connects
- ServiceName - the name of service the client requests

Results

Service returned.

Example

ASCII Version

```
Server *server= NULL;
Service *service = NULL;
//Create Server
server = createServer();
...
// get Service From Server
service = getServiceFromServer(server,"ValidateAddress" );
```

Unicode Version

```
// construct 16-bit string
UChar serviceName[32];
```

```
char* sName="ValidateAddress";
u_charsToUChars(sName, serviceName, strlen(sName));
serviceName [ strlen(sName)]=0;
service = getServiceFromServer(server , serviceName );
```

Service

The Service structure is used to process the message (in other words, it sends the message to the server and receives a response from the server). The following table summarizes the functions performed in the Service structure.

ProcessMessage

Processes the input message and retrieves the response message from the server.

Note: You will need to call DeleteMessage() to free memory when this returned message is no longer used.

Syntax

```
int processMessage (Service* service, Message* request, Message* returnVal)
```

Parameters

- Service—the service the client requests.
- Request—the input message which contains the "option" setting and the dataset.
- returnVal—returns the response message from the server.

Results

Returns 0 (if successful) or error code.

Example

```
Message *request = NULL;
Message *reply = NULL;
int nRet;
 ...
 // Assume that service is given here
 // Create Input Message
 request = createMessage();
 ... more code to fill dataTable information in request message
//Process Input Message, return output Message
 nRet = processMessage(service, request, &reply);
 if(nRet != SUCCESSFUL_RETURN)
 {
 printf("Error Occurred, " );
 printf(getErrorMessage(nRet));
 return ;
 }
 if(request)
 nRet = deleteMessage(request);
 if(reply)
 nRet = deleteMessage(reply);
```

Message

The Message structure sends your input data and receives your output data from the service. The properties for Message include context properties, such as account ID, account password, service name, and service method; option properties, which are the service-specific runtime options; and error properties, which are the error class, error message and error stacktrace.

CreateMessage

Creates a message.

Syntax

```
Message*      createMessage()
```

Parameters

None.

Results

The message created.

Example

```
Message* request = NULL;
request = createMessage();
```

DeleteMessage

Deletes the message.

Syntax

```
int deleteMessage(Message* message)
```

Parameters

- Message— the message to be deleted

Results

Returns 0 if successful or error code.

Example

```
int nRet = deleteMessage(message);
```

GetContext

Gets the value of the context entity identified by the name in the context session of the message. "Context" entities include the following constants: account id, account password, service name, service method.

Syntax

ASCII Version

```
const char* getContext(Message* message,const char* name)
```

Unicode Version

```
const UChar * getContext(Message* message,const UChar* name)
```

Parameters

- Message - the message to which this function applies
- Name - the name whose associated value is to be returned

Result

Returns the value for the name in the context entity. If the name does not exist, the method returns empty string.

Example

ASCII Version

```
const char* value = getContext(message, "account.id");
```

Unicode Version

```
UChar* value;
// construct 16-bit string
UChar accountID[32];
char* account="account.id";
u_charsToUChars(account, accountID, strlen(account));
accountID[ strlen(account)]=0;
value = getContext(message, accountID);
```

GetContextMap

Gets the Map that contains all of the context entries.

Syntax

ASCII Version

```
MAP_STRING**getContextMap(Message* message)
Where the MAP_STRING is defined by
typedef struct map_string{
char* key;
char* value;
}MAP_STRING;
```

Unicode Version

```
MAP_STRING**getContextMap(Message* message)
Where the MAP_STRING is defined by
typedef struct map_string{
UChar* key;
UChar* value;
}MAP_STRING;
```

Parameters

- Message - the message to which this function applies

Results

Returns the array of MAP_STRING that contains all of the context entries.

Example

ASCII Version

```
int i;
char* name;
char* value;
MAP_STRING** mapping;
mapping = getContextMap( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

Unicode Version

```
int i;
UChar* name;
UChar* value;
MAP_STRING** mapping;
mapping = getContextMap( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

PutContext

Sets the value for given name in the "context" properties. If there is an existing value present for the entity identified by the name, it is replaced. Context properties include the following constants: account ID, account password, service name, service key, and request ID.

Syntax

ASCII Version

```
int putContext(Message* message,const char* name,
const char* value)
```

Unicode Version

```
int putContext(Message* message,const UChar* name,
const UChar* value)
```

Parameters

- Message—message to which this function applies
- Name—name with which the specified value is to be associated
- Value—value to be associated with the specified name

Results

Returns 0 (if successful) or error code.

Example

ASCII Version

```
int nRet;
Message* message = createMessage();
nRet = putContext( message, "account.id", "user1") ;
```

Unicode Version

```
int nRet;
Message* message;
// construct 16-bit string
UChar accountID[32];
char* account="account.id";
UChar accountIDValue[32];
char* accountValue="user1";
u_charsToUChars(account, accountID, strlen(account));
accountID [ strlen(account)]=0;
u_charsToUChars(accountValue, accountIDValue, strlen(accountValue));
accountIDValue [ strlen(accountValue)]=0;
message = createMessage();
nRet = putContext( message, accountID, accountIDValue);
```

PutContextMap

Adds the new context properties to the current context properties.

Syntax

```
int putContextMap(Message* message, MAP_STRING** context)
```

Parameters

- Message - the message to which this function applies
- The new context map to be added to the current context map.

Results

Returns 0 (if successful) or error code.

Example

ASCII Version

```
MAP_STRING** mapping;
Message* message;
message = createMessage();
int nRet;
mapping = (MAP_STRING **)malloc(3 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet = putContextMap( message, mapping) ;
```

Unicode Version

```
MAP_STRING** mapping;
Message* message;
int nRet;
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";

u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;

message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet = putContextMap( message, mapping) ;
```

SetContextMap

Overwrites the current context properties with the new context properties.

Syntax

```
int setContextMap(Message* message, MAP_STRING** context)
```

Parameters

- Message - the message to which this function applies
- The new context map to be used to replace the current context map.

Results

Returns 0 (if successful) or error code.

Example

ASCII Version

```
MAP_STRING** mapping;
Message* message;
int nRet;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet=setContextMap( message, mapping) ;
```

Unicode Version

```
MAP_STRING** mapping;
Message* message;
int nRet;
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";
u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1 ;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet=setContextMap( message, mapping) ;
```

GetOption

Gets the value of the option entity identified by the name in the option session of the message. "Options" entities include the service-specific runtime options, such as output casing, output data format, and so on.

Syntax

ASCII Version

```
const char* getOption(Message* message,const char* name)
```

Unicode Version

```
const Uchar* getOption(Message* message,const UChar* name)
```

Parameters

- Message - the message to which this function applies
- Name- the name whose associated value is to be returned

Results

Returns the value for the name in the "option" property in the message or an empty string if the name does not exist.

Example

ASCII Version

```
const char* value = getOption (message, " OutputCasing");
```

Unicode Version

```
UChar* value;
// construct 16-bit string
UChar option[32];
char* optionValue="OutputCasing";
u_charsToUChars(optionValue, option, strlen(optionValue));
option [ strlen(optionValue)]=0;
value = getOption(message, option);
```

GetOptions

Gets the map that contains all of the option entries.

Syntax

```
MAP_STRING**        getOptions(Message* message)
```

Parameters

- Message—the message to which this function applies

Results

Returns the array of MAP_STRING that contains all of the context entries.

Example

ASCII Version

```
int i;
char* name;
char* value;
MAP_STRING** mapping;
mapping = getOptions( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

Unicode Version

```
int i;
UChar* name;
UChar* value;
MAP_STRING** mapping;
mapping = getOptions( message);
i=0;
while(mapping[i] != NULL)
{
name= mapping[i]->key;
value = mapping[i]->value;
i++;
}
```

PutOption

Sets the value for given name in the "option" properties. If there is an existing value present for the entity identified by the name, it is replaced. Option properties are the service-specific run-time options.

Syntax

ASCII Version

```
int putOption(Message* message,const char* name,
const char* value)
```

Unicode Version

```
int putOption(Message* message,const UChar* name,
const UChar* value)
```

Parameters

- Message - the message to which this function applies
- Name - with which the specified value is to be associated.
- Value - to be associated with the specified name.

Results

Returns 0 (if successful) or error code.

Example

ASCII Version

```
int nRet;
Message* message = createMessage();
nRet = putOption( message, "OutputCasing", "M");
```

Unicode Version

```
int nRet;
Message* message;
// construct 16-bit string
UChar option[32];
char* optionString="OutputCasing";

UChar optionValue[32];
char* optionValueString="M";

u_charsToUChars(optionString, option, strlen(optionString));
option[ strlen(optionString)]=0;

u_charsToUChars(optionValueString, optionValue,
strlen(optionValueString));
optionValue [ strlen(optionValueString)]=0;

message = createMessage();
nRet = putOption( message, option, optionValue);
```

PutOptions

Adds the new option properties to the current option properties.

Syntax

```
int  putOptions(Message* message, MAP_STRING** context)
```

Parameters

- Message - the message to which this function applies
- The new option map to be added to the current option properties

Results

Returns 0 if successful or error code.

Example

ASCII Version

```
MAP_STRING** mapping;
Message* message;
message = createMessage();
int nRet;
mapping = (MAP_STRING **)malloc(3 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet = putOptions( message, mapping) ;
```

Unicode Version

```
MAP_STRING** mapping;
Message* message;
int nRet;
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";
u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet = putOptions ( message, mapping) ;
```

SetOptions

Overwrites the current option properties with the new option properties.

Syntax

```
int  setOptions(Message* message, MAP_STRING** context)
```

Parameters

- Message - the message to which this function applies
- The new option map to be used to replace the current option map

Results

Returns 0 if successful or error code.

Example

ASCII Version

```
MAP_STRING** mapping;
Message* message;
int nRet;
message = createMessage();
mapping = (MAP_STRING **)malloc(3 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = "key1" ;
mapping[0]->value = "value1" ;
mapping[1] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[1]->key = "key2" ;
mapping[1]->value = "value2" ;
mapping[2] = NULL;
nRet=setOptions( message, mapping) ;
```

Unicode Version

```
MAP_STRING** mapping;
Message* message;
int nRet;
UChar key1[32];
char* key1String="key1";
UChar value1[32];
char* value1String="value1";
u_charsToUChars(key1String, key1, strlen(key1String));
key1[ strlen(key1String)]=0;
u_charsToUChars(value1String, value1, strlen(value1String));
value1[ strlen(value1String)]=0;
message = createMessage();
mapping = (MAP_STRING **)malloc(2 * sizeof(MAP_STRING *));
mapping[0] = (MAP_STRING *)malloc( sizeof(MAP_STRING));
mapping[0]->key = key1 ;
mapping[0]->value = value1 ;
mapping[1] = NULL;
nRet= setOptions ( message, mapping) ;
```

GetError

Gets the error message from the message.

Syntax

ASCII Version

```
const char* getError(Message* message )
```

Unicode Version

```
const UChar* getError(Message* message )
```

Parameters

- Message - the message to which this function applies

Result

Returns the error message in message.

Example

ASCII Version

```
const char* error = getError(message );
```

Unicode Version

```
const UChar* error = getError(message );
```

GetDataTable

Gets the DataTable in the message.

Syntax

```
DataTable*   getDataTable(Message* message )
```

Parameters

- Message - the message to which this function applies

Example

```
// Assume that message is given here
DataTable  *dataTable   ;
dataTable = getDataTable( message );
```

DataTable

DataTable contains the records for the input and output data.

CreateDataTable

Create the DataTable.

Syntax

```
DataTable* createDataTable()
```

Results

Returns the DataTable created.

Example

```
DataTable* dataTable;
dataTable = createDataTable();
```

DeleteDataTable

Deletes the DataTable.

Syntax

```
int deleteDataTable(DataTable* dataTable)
```

Parameters

- Datatable - the DataTable to be deleted

Example

```
DataTable* dataTable;
dataTable = createDataTable();
...
if(dataTable) deleteDataTable(dataTable);
```

AddColumn

Adds the new column.

Syntax

ASCII Version

```
int addColumn(DataTable* dataTable, const char* columnName,
int* indexReturn)
```

Unicode Version

```
int addColumn(DataTable* dataTable, const UChar* columnName,
int* indexReturn)
```

Parameters

- Datatable - the DataTable to which this function applies
- Column name to be added to the DataTable
- Returns the corresponding index

Results

Returns 0 if successful or error code.

Exceptions

- Blank column name
- Duplicate column name

Example

ASCII Version

```
int nIndex;
int nRet;
nRet= addColumn( dataTable, "AddressLine1", &nIndex);
nRet= addColumn( dataTable, "City",&nIndex);
nRet= addColumn( dataTable, "State", &nIndex);
if(nRet != SUCCESSFUL_RETURN)
{
printf(getErrorMessage(nRet));
return ;
}
```

Unicode Version

```
int nRet;
int nIndex;
UChar* error;
UChar city[64];
char* cityString= "City"
u_charsToUChars(cityString, city, strlen(cityString));
city[ strlen(cityString)]=0;

nRet= addColumn( dataTable, city,&nIndex);
if(nRet != SUCCESSFUL_RETURN)
{
error = getErrorMessage(nRet);
//more code
}
```

GetColumnNames

Gets all the column names.

Syntax

ASCII Version

```
char** getColumnNames(dataTable )
```

Unicode Version

```
UChar** getColumnNames(dataTable )
```

Parameters

- Datatable - the DataTable to which this function applies

Results

Returns the array of column names.

Example

ASCII Version

```
char* value;
char** columnNames;
int i;
columnNames =getColumnNames ( dataTable) ;
for( i=0; i < getColumncount( dataTable); i++)
```

```
{
value = columnNames[i];
}
```

Unicode Version

```
UChar* value;
UChar** columnNames;
int i;
columnNames =getColumnNames ( dataTable) ;
for( i=0; i < getColumncount( dataTable); i++)
{
value = columnNames[i];
}
```

GetColumnIndex

Gets the corresponding column index.

Syntax

ASCII Version

```
int getColumnIndex(DataTable* dataTable ,const char* columnName)
```

Unicode Version

```
int getColumnIndex(DataTable* dataTable ,const UChar* columnName)
```

Parameters

- Datatable - the DataTable to which this function applies
- Column name

Results

Returns the corresponding column index.

Example

ASCII Version

```
int nIndex ;
nIndex = getColumnIndex(dataTable ,"AddressLine1")
```

Unicode Version

```
int nIndex ;
UChar columnName[64];
char* columnNameStr= "AddressLine1" u_charsToUChars(columnNameStr,
columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
nIndex = getColumnIndex(dataTable , columnName);
```

GetColumnCount

Gets the number of columns.

Syntax

```
int getColumnCount(DataTable* dataTable )
```

Parameters

- Datatable - the DataTable to which this function applies

Results

Returns the number of columns.

Example

```
// Assume that dataTable is given here int nColumnCount ;
nColumnCount = getColumnCount( dataTable ) ;
```

Clear

Clears the data in DataTable.

Syntax

```
int clear(DataTable* dataTable)
```

Parameters

- Datatable - the DataTable to which this function applies

Results

Returns 0 if successful or error code.

Example

```
// Assume that dataTable is given here
clear(dataTable);
```

GetDataRows

Gets an array of all DataRows in the DataTable.

Syntax

```
DataRow** getDataRows(DataTable* dataTable)
```

Parameters

- Datatable - the DataTable to which this function applies

Results

Returns an array of DataRows.

Example

```
// Assume that dataTable is given here
DataRows** rows;
DataRow* dataRow;
int i;
int j;
rows = getDataRows( dataTable);
for( i=0; i < getRowCount( dataTable); i++)
{
 dataRow =  rows[i];

 for(j=0; j < getColumnCount(dataTable); j++)
 {
 value = (char*)getByIndex( dataRow, j);
 }
}
```

AddRow

Adds a DataRow to the DataTable.

Syntax

```
int addRow(DataTable* dataTable, DataRow* dataRow)
```

Parameter

- Datatable - the DataTable to which this function applies
- Datarow to be added to the DataTable

Results

Returns 0 if successful or error code.

Example

```
// Assume that dataTable is given here DataRow* newDataRow;
int nRet;
newDataRow = newRow( dataTable );
setByIndex (newDataRow, 0 , "10535 Boyer Blvd");
setByIndex (newDataRow, 1 , "Austin");
setByIndex (newDataRow, 2 , "Texas");
nRet = addRow( dataTable, newDataRow);
```

NewRow

Creates a new DataRow in the DataTable.

Syntax

```
DataRow* newRow(DataTable* dataTable )
```

Parameter

- Datatable - the DataTable to which this function applies

Results

Returns the new created DataRow.

Example

```
// Assume that dataTable is given here
DataRow* newDataRow;
int nRet;
newDataRow = newRow( dataTable );
setByIndex (newDataRow, 0 , "10535 Boyer Blvd");
setByIndex (newDataRow, 1 , "Austin");
setByIndex (newDataRow, 2 , "Texas");
nRet = addRow( dataTable, newDataRow);
```

GetRowCount

Gets the number of the DataRows in this DataTable.

Syntax

```
int getRowCount(DataTable* dataTable)
```

Parameter

- Datatable - the DataTable to which this function applies

Results

Returns the number of the DataRows in this DataTable.

Example

```
// Assume that dataTable is given here int nRowCount ;
nRowCount = getRowCount( dataTable);
```

MergeDataTable

Merges the given DataTable and the current DataTable.

Syntax

```
int mergeDataTable(DataTable* dataTable ,DataTable* other )
```

Parameter

- Datatable - the DataTable to which this function applies
- Other DataTable to be merged with the current DataTable

Results

Returns 0 if successful or error code.

Example

```
// Assume that dataTable and otherDataTable are given here
mergeDataTable (dataTable ,otherDataTableDataRow)
```

DataRow

DataRow contains the record for the input and output data.

CreateDataRow

Creates the DataRow.

Syntax

```
DataRow*   createDataRow()
```

Results

Returns the DataRow created.

Example

```
DataRow* dataRow;
dataRow = createDataRow();
```

DeleteDataRow

Deletes the DataRow.

Syntax

```
int   deleteDataRow(DataRow* dataRow)
```

Parameter

- The DataRow to be deleted

Example

```
DataRow* dataRow;
dataRow = createDataRow();
...
if(dataRow)
  deleteDataRow (dataRow);
```

GetColumnNamesFromRow

Gets all the column names.

Syntax

ASCII Version

```
char** getColumnNamesFromRow(DataRow* dataRow)
```

Unicode Version

```
UChar** getColumnNamesFromRow(DataRow* dataRow)
```

Parameter

- Datarow - the DataRow to which this function applies

Results

Returns the array of column names.

Example

ASCII Version

```
char* value;
char** columnNames;
int i;
columnNames = getColumnNamesFromRow (dataRow) ;
for( i=0; i < getColumnCountFromRow (dataRow); i++)
{
 value = columnNames[i];
}
```

Unicode Version

```
UChar* value;
UChar** columnNames;
int i;
columnNames = getColumnNamesFromRow (dataRow) ;
for( i=0; i < getColumnCountFromRow (dataRow); i++)
{
 value = columnNames[i];
}
```

GetColumnIndexFromRow

Gets the corresponding column index.

Syntax

ASCII Version

```
int getColumnIndexFromRow(DataRow* dataRow, const char* name)
```

Unicode Version

```
int getColumnIndexFromRow(DataRow* dataRow, const UChar* name)
```

Parameter

- Datarow - the DataRow to which this function applies
- Column name

Results

Returns the corresponding column index.

Example

ASCII Version

```
int nIndex
nIndex = getColumnIndexFromRow ("AddressLine1");
```

Unicode Version

```
int nIndex
UChar columnName[64];
char* columnNameStr= "AddressLine1"
u_charsToUChars(columnNameStr, columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
nIndex = getColumnIndexFromRow (columnName);
```

GetColumnCountFromRow

Gets the number of columns.

Syntax

```
int  getColumnCountFromRow(DataRow* dataRow )
```

Parameter

- Datarow - the DataRow to which this function applies

Results

Returns the number of columns.

Example

```
//Assume that the dataRow is given here
int nColumnCount ;
nColumnCount = getColumnCountFromRow (dataRow );
```

GetByIndex

Gets the value from the fields array by the column index in this DataRow.

Syntax

ASCII Version

```
const char* getByIndex(DataRow* dataRow, int index)
```

Unicode Version

```
const UChar* getByIndex(DataRow* dataRow, int index)
```

Parameter

- Datarow - the DataRow to which this function applies
- Index with which the specified value is to be associated.

Results

Returns the value for the column index in the DataRow, returns empty string if the index in invalid.

Example

ASCII Version

```
char* value = getByIndex( dataRow, 0);
```

Unicode Version

```
UChar* value = getByIndex( dataRow, 0);
```

GetByName

Gets the value from the fields array by the column name in this DataRow.

Syntax

ASCII Version

```
const char* getByName(DataRow* dataRow, const char* name )
```

Unicode Version

```
const UChar* getByName(DataRow* dataRow, const UChar* name )
```

Parameter

- Datarow - the DataRow to which this function applies
- Name with which the specified value is to be associated

Results

Returns the value for the column name in the DataRow, returns empty string if the column name does not exist.

Example

ASCII Version

```
char* value = getByName ( dataRow, "City")
```

Unicode Version

```
UChar* value;
UChar columnName[64];
char* columnNameStr= "City"
u_charsToUChars(columnNameStr, columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
value = getByName ( dataRow, columnName);
```

MergeDataRow

Merges the given DataRow and the current DataRow.

Syntax

```
int   mergeDataRow(DataRow* dataRow, DataRow* other)
```

Parameter

- Datarow - the DataRow to which this function applies
- Other DataRow to be merged with the current DataRow

Results

Returns 0 if successful or error code.

Example

```
//Assume that the dataRow and otherDataRow are given here
int nRet;
nRet= mergeDataRow(dataRow, otherDataRow);
```

SetByName

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

ASCII Version

```
int setByName(DataRow* dataRow, const char* name, const char* value)
```

Unicode Version

```
int setByName(DataRow* dataRow, const UChar* name, const
UChar* value)
```

Parameters

- Datarow - the DataRow to which this function applies
- Name with which the specified value is to be associated
- Value to be associated with the specified name

Exceptions

If input Blank column name or Duplicate column name, return error

Results

Returns 0 if successful or error code.

Example

ASCII Version

```
int nRet;
nRet= setByName (dataRow, "City", "Austin");
if(nRet != SUCCESSFUL_RETURN)
{ printf(getErrorMessage(nRet));
//more code
}
```

Unicode Version

```
int nRet;
UChar* error;
UChar columnName[64];
char* columnNameStr= "City"
UChar columnValue[64];
char* columnValueStr= "Austin";
u_charsToUChars(columnNameStr, columnName, strlen(columnNameStr));
columnName [strlen(columnNameStr)]=0;
u_charsToUChars(columnValueStr, columnValue, strlen(columnValueStr));
columnValue [strlen(columnValueStr)]=0;
nRet= setByName (dataRow, columnName, columnValue);
if(nRet != SUCCESSFUL_RETURN)
{ error = getErrorMessage(nRet);
//more code
}
```

SetByIndex

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

ASCII Version

```
int setByIndex(DataRow* dataRow, int index, const char* value)
```

Unicode Version

```
int setByIndex(DataRow* dataRow, int index, const UChar* value)
```

Parameters

- Datarow - the DataRow to which this function applies
- Column index with which the specified value is to be associated
- Value to be associated with the specified name

Exceptions

- The column index is invalid

Results

Returns 0 if successful or error code.

Example

ASCII Version

```
int nRet;
nRet= setByIndex (dataRow, 1, "Austin");
if(nRet != SUCCESSFUL_RETURN)
{
printf(getErrorMessage(nRet));
 //more code
}
```

Unicode Version

```
int nRet;
UChar* error;
UChar columnValue[64];
char* columnValueStr= "Austin";
u_charsToUChars(columnValueStr, columnValue, strlen(columnValueStr));
columnValue [strlen(columnValueStr)]=0;
nRet= setByIndex (dataRow, 1, columnValue);
if(nRet != SUCCESSFUL_RETURN)
{
error = getErrorMessage(nRet);
//more code
}
```

AddChild

Adds a new DataRow to the named parent/child relationship. If the named relationship exists, the supplied DataRow will be appended to the existing DataRow collection. Otherwise, a new collection will be created with the supplied DataRow as its only element.

Syntax

ASCII Version

```
void addChild(DataRow* dataRow, const char* childName, DataRow* childDataRow)
```

Unicode Version

```
void addChild(DataRow* dataRow, const UChar* childName, DataRow* childDataRow)
```

Parameters

- The name of the parent/child relationship (e.g., "Flood Plain Data," "References," "Used By," and so forth)
- The DataRow to be added to the relationship

Example

ASCII Version

```
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();

setByName(child1DataRow1, "City", "Austin");
setByName(child1DataRow1, "State", "Texas");

addChild( dataRow, "child1", child1DataRow1);
```

Unicode Version

```
UChar* convertcharToUChar( char* name, UChar* value)
 {
 int lenName= strlen(name);

 u_charsToUChars(name, value, lenName );

 value[ lenName]=0;
 return value;
 } >
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();
UChar     name[128];
UChar     columnValue[128];
setByName(child1DataRow1, convertcharToUChar("City", name),
  convertcharToUChar("Austin", columnValue));
setByName(child1DataRow1, convertcharToUChar("State", name),
  convertcharToUChar("Texas", columnValue));
addChild( dataRow, "child1", child1DataRow1);
```

GetChildren

Retrieves the child rows from a named relationship.

Syntax

ASCII Version

```
DataRow** getChildren(DataRow* dataRow,const char* childName)
```

Unicode Version

```
DataRow** getChildren(DataRow* dataRow,const UChar* childName)
```

Parameters

- The name of the parent/child relationship, for example "Flood Plain Data", "References", "Used By", and so forth.

Results

Returns the child rows from the named relationship.

Example

ASCII Version

```
DataRow** child1Rows;
child1Rows = getChildren(dataRow, "child1");
```

Unicode Version

```
DataRow** child1Rows;
UChar childName[128];
/* see convertcharToUChar in the Example section of "addChild" */
child1Rows = getChildren(dataRow, convertcharToUChar("child1", childName));
```

ListChildNames

Retrieves all of the names of the named parent/child relationships.

Syntax

ASCII Version

```
char** listChildNames(DataRow* dataRow)
```

Unicode Version

```
UChar** listChildNames(DataRow* dataRow)
```

Results

Returns the set of the names of the named parent/child relationships.

Example

ASCII Version

```
char** childsNames;
childsNames =listChildNames( dataRow);
```

Unicode Version

```
UChar** childsNames;
childsNames=listChildNames( dataRow);
```

SetChildren

Sets the rows of a supplied, named parent/child relationship. If rows previously existed under this name, they will be returned to the caller.

Syntax

ASCII Version

```
DataRow** setChildren(DataRow* dataRow, const char* childName, DataRow**
dataRows)
```

Unicode Version

```
DataRow** setChildren(DataRow* dataRow, const UChar* childName, DataRow**
 dataRows)
```

Results

Returns the set of the names of the named parent/child relationships.

Example

ASCII Version

```
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();
DataRow* child1DataRow2 = createDataRow();
DataRow* child2DataRow = createDataRow();
```

```
DataRow** child2Rows;
DataRow** returnRows;

setByName(child1DataRow1, "Address", "200 Congress");
setByName(child1DataRow1, "City", "Austin");

setByName(child1DataRow2, "Address", "100 Congress");
setByName(child1DataRow2, "City", "Dallas");

setByName(child2DataRow, "Address", "100 Congress");
setByName(child2DataRow, "City", "Austin");

addChild( dataRow, "child1", child1DataRow1);
addChild( dataRow, "child1", child1DataRow2);
addChild( dataRow, "child2", child2DataRow );

child2Rows=getChildren(dataRow, "child2");

returnRows=setChildren( dataRow, "child1", child2Rows);
```

Unicode Version

```
DataRow* dataRow = createDataRow();
DataRow* child1DataRow1 = createDataRow();
DataRow* child1DataRow2 = createDataRow();
DataRow* child2DataRow = createDataRow();
DataRow** child2Rows;
DataRow** returnRows;
UChar name[128];
UChar columnValue[128];
UChar childName[128];

setByName(child1DataRow1, convertcharToUChar("Address", name),
convertcharToUChar("200 Congress", columnValue));
setByName(child1DataRow1, convertcharToUChar("City", name),
convertcharToUChar("Austin", columnValue));
setByName(child1DataRow2, convertcharToUChar("Address", name),
convertcharToUChar("100 Congress", columnValue));
setByName(child1DataRow2, convertcharToUChar("City", name)
convertcharToUChar("Dallas", columnValue) );
setByName(child2DataRow, convertcharToUChar("Address", name),
convertcharToUChar("100 Congress", columnValue) );
setByName(child2DataRow, convertcharToUChar("City", name),
convertcharToUChar("Austin", columnValue) );

addChild( dataRow, convertcharToUChar("child1", childName), child1DataRow1);
addChild( dataRow, convertcharToUChar("child1", childName),child1DataRow2);
addChild( dataRow, convertcharToUChar("child2", childName), child2DataRow
 );

child2Rows=getChildren(dataRow, convertcharToUChar("child2",childName));
returnRows=setChildren( dataRow, convertcharToUChar("child1",childName),
child2Rows);
```

3

The C++ API

In this section:

- Introduction to the C++ API 64
- Server 72
- Service 75
- Message 76
- DataTable 84
- DataRow 89

Introduction to the C++ API

The C++ API consists of the following classes:

- Server
- Service
- Message
- DataTable
- DataRow

UnicodeString in ICU is a string class that stores Unicode characters directly and provides similar functionality as the Java String and StringBuffer classes. The Spectrum™ Technology Platform Unicode C++ API uses this class to store Unicode strings.

Supported Libraries

Spectrum™ Technology Platform provides an ASCII and Unicode version C API, while the Unicode version remains as compatible as possible with the original ASCII-version API design. Spectrum™ Technology Platform applies International Components for Unicode (ICU) in the API to support the Unicode feature. ICU is a mature, widely used set of C/C++ libraries for Unicode support and is developed by IBM.

The Unicode standard defines a default encoding based on 16-bit code units. This is supported in ICU by the definition of the UChar to be an unsigned 16-bit integer type(unsigned short *). This is the base type for character arrays for strings in ICU. Spectrum™ Technology Platform uses UChar as the Unicode string representation in our C API.

Note: Not all services support the full Unicode character set. For example, the ValidateAddress service supports the ISO 8859-1 character set for US inputs and International inputs and outputs and the CP 850 character set for Canadian inputs and outputs. However, the Unicode libraries should be used whenever your input data may contain any non-ASCII character, even if the underlying service does not support the full Unicode character set.

For detailed information about UChar, please refer to the following two sites:

- icu.sourceforge.net/userguide/
- www-306.ibm.com/software/globalization/icu/index.jsp

Windows

Each API configuration produces library files with a common base name (g1client) but with a unique suffix and possibly prefix ("lib" in the case of static libraries). The library suffixes work like this:

```
<lib>g1client<S><U><D>.<lib|dll>
```

- lib—indicates a static library.
- dll—indiates a dynamic (shared) library.
- S—indicates a single-threaded build. If this is absent it indicates a multi-threaded build.
- U—indicates a UNICODE version build. If this is absent it indicates an ASCII build.
- D—indicates a debug build. If this suffix is absent it indicates an optimized release build.

To enable the UNICODE version, the LIB_UNICODE macro definition must be in your project.

To use the static C/C++ API library UNICODE version, you need to define U_STATIC_IMPLEMENTATION in your project.

To use the dynamic version, you need to define G1CLIENT_DLL in your project.

We also provide a file called "auto_link.h" in the header file directory and it automatically links to all the corresponding libraries according to the project settings.

To call 64-bit libraries in Windows, you need to define VER_64 in your project.

Static Library

Note: The names provided in this section are for 32-bit libraries. For 64-bit libraries, replace "32" with "64" in the library name.

Single Threaded/Release		
	ASCII	Unicode
g1	libg1client_S.lib	libg1client_SU.lib
openssl	otlibeay32.lib otlibssl32.lib	otlibeay32.lib otlibssl32.lib
opentop	opentop.lib	opentopw.lib
icu		libicuuc.lib libicudt.lib libicuin.lib libicuio.lib
Poco	PocoXML32.lib	PocoXML32w.lib
Single Threaded/Debug		
	ASCII	Unicode
g1	libg1client_SD.lib	libg1client_SUD.lib
openssl	otlibeay32d.lib otlibssl32d.lib	otlibeay32d.lib otlibssl32d.lib
opentop	opentopd.lib	opentopwd.lib
icu		libicuucd.lib libicudtd.lib libicuind.lib libicuiod.lib
Poco	PocoXML32d.lib	PocoXML32wd.lib
Multi/Release (using Multi-Threaded CRT)		
	ASCII	Unicode
g1	libg1client.lib	libg1client_U.lib
openssl	otlibeay32mt.lib otlibssl32mt.lib	otlibeay32mt.lib otlibssl32mt.lib
opentop	opentopmt.lib	opentopmtw.lib
icu		libicuucmt.lib libicudtmt.lib libicuinmt.lib libicuiomt.lib
Poco	PocoXMLmt32.lib	PocoXML32mtw.lib
Multi/Debug (using Multi-Threaded CRT)		
	ASCII	Unicode
g1	libg1client_D.lib	libg1client_UD.lib
openssl	otlibeay32mtd.lib otlibssl32mtd.lib	otlibeay32mtd.lib otlibssl32mtd.lib
opentop	opentopmtd.lib	opentopmtwd.lib

icu		libicuucmtd.lib libicudtmtd.lib libicuinmtd.lib libicuiomtd.lib
Poco	PocoXMLmt32d.lib	PocoXML32mtwd.lib

Dynamic Library

Note: The names provided in this section are for 32-bit libraries. For 64-bit libraries, replace "32" with "64" in the library name.

Multi/Release (using Multi-Threaded CRT)		
	ASCII	Unicode
g1	g1client.dll	g1client_U.dll
openssl	otlibeay32mts.dll otlibssl32mts.dll	otlibeay32mts.dll otlibssl32mts.dll
opentop	opentopmts.dll	opentopmtws.dll
icu		icuuc32.dll icuio32.dll icuin32.dll icudt32.dll
Poco	PocoXML32mts.dll	PocoXML32mtws.dll
Multi/Debug (using Multi-Threaded CRT)		
	ASCII	Unicode
g1	g1client_D.dll	g1client_UD.dll
openssl	otlibeay32mtds.dll otlibssl32mtds.dll	otlibeay32mtds.dll otlibssl32mtds.dll
opentop	opentopmtds.dll	opentopmtwds.dll
icu		icuuc32d.dll icuio32d.dll icuin32d.dll icudt32d.dll
Poco	PocoXML32mtds.dll	PocoXML32mtwds.dll

Unix

Each ClientSDK configuration produces library files with a common base name (libg1client) but with a unique suffix. Spectrum™ Technology Platform provides a multithread and release build for ASCII version and UNICODE version.

The library suffixes work like this:-

```
libg1client<U>.<so|sl|a>
```

- U—indicates a UNICODE version build. If this is absent it indicates an ASCII build.

To use the UNICODE version, you need to define LIB_UNICODE in your project.

In UNICODE Version C++ API, the namespace for all classes is g1client.

AIX		
	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libssl.so	libcrypto.so libssl.so

opentop	libopentop-xlCmt.so	libopentop-xlCmtw.so libotxml-xlCmtw.so
icu		libicudata34.a libicui18n34.a libicuio34.a libicuuc34.a
Poco	libPocoXML.so	
HP-UX		
	ASCII	Unicode
g1	libg1client.sl	libg1client_U.sl
openssl	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7	libcrypto.sl libssl.sl libcrypto.sl.0.9.7 libssl.sl.0.9.7
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl
icu		libicudata.sl libicudata.sl.34 libicui18n.sl libicui18n.sl.34 libicuio.sl libicuio.sl.34 libicuuc.sl libicuuc.sl.34
Poco	libPocoXML.sl	
Itanium		
	ASCII	Unicode
g1	libg1client.sl	libg1client_U.sl
openssl	libcrypto.a libssl.a	libcrypto.a libssl.a
opentop	libopentop-accmt.sl	libopentop-accmtw.sl libotxml-accmtw.sl
icu		libicudata.sl libicudata.sl.34 libicudata.sl.34.0 libicui18n.sl libicui18n.sl.34 libicui18n.sl.34.0 libicuio.sl libicuio.sl.34 libicuio.sl.34.0 libicuuc.sl libicuuc.sl.34 libicuuc.sl.34.0
Poco	libPocoXML.sl	
Linux		
	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-gccmt.so	libopentop-gccmtw.so libotxml-gccmtw.so
icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34

		libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34
Poco	libPocoXML.so	
Solaris		
	ASCII	Unicode
g1	libg1client.so	libg1client_U.so
openssl	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7	libcrypto.so libcrypto.so.0.9.7 libssl.so libssl.so.0.9.7
opentop	libopentop-fortemt.so	libopentop-fortemtw.so libotxml-fortemtw.so
icu		libicudata.so libicudata.so.34 libicui18n.so libicui18n.so.34 libicuio.so libicuio.so.34 libicuuc.so libicuuc.so.34
Poco	libPocoXML.so	

Constants

The C++ API uses two sets of constants. The first set is for the Server class described in the table below.

Table 14: Constants for the Server Component

Constant Name	Description/Default	Example
Server::HOST	String for server host name. Default is "localhost".	65.89.200.89
Server::PORT	String for server port. Default is "8080".	10119
Server::ACCOUNT_ID	String for server account ID. No default value.	user1
Server::ACCOUNT_PASSWORD	String for server account password. No default value.	user1
Server::CONNECTION_TIMEOUT	String for server connection timeout, in milliseconds. Default is "5000".	50000
Server::CONNECTION_TYPE	String for server connection type. Currently only supports HTTP, HTTPS, or SOCKET. Default is "HTTP".	HTTP(S)
Server::PROXY_HOST	String for proxy server host name. No default value.	192.168.1.77
Server::PROXY_PORT	String for proxy server port. No default value.	8080

Constant Name	Description/Default	Example
Server::PROXY_USER	String for proxy server account ID. No default value.	user1
Server::PROXY_PASSWORD	String for proxy server account password. No default value.	user1

The second set of constants is for the Message class:

Table 15: Constants for the Message Component

Constant Name	Description	Example
Message::CONTEXT_ACCOUNT_ID	String for message context account ID.	user1
Message::CONTEXT_ACCOUNT_PASSWORD	String for message context account password.	user1
Message::CONTEXT_SERVICE_NAME	String for message context service name.	echoservice

Error Messages

In order to get error messages, use the Exception class. Use the try/catch constructs to capture the error message. For example:

```
try{
  Server *server=new Server();

  //Connect to server
  server->connect();

  }catch(Exception e)
  {
   // ASCII Version-use the following code
  cout << "Error Occurs," << e.getErrorMessage();
  //Unicode Version -use the following code

  UnicodeString error = e.getErrorMessage() ;

  wcout << error.getTerminatedBuffer();
  }
```

The C++ API uses the following error messages:

- Error Messages for Connection:
  - "Connection type not supported"
  - "Client timeout"
  - "Blank connection property name"
  - "Blank property name"
- Error Messages for creating DataTable:
  - "Blank column name"
  - "Duplicated column name"

- "The column index is invalid"

- Error Messages for MessagePackaging Exception:
  - "Input Message is null"
  - "Failed to connect to Server"
  - "Failed to disconnect from Server"
  - "Failed to open Http Connection"
  - "Failed to get Service"
  - "Failed to package the message using Serializer and Encoding"

SmartPointer

Spectrum™ Technology Platform provides a class called SmartPointer that uses a simple form of reference counting to help track allocation of dynamic memory and perform memory management task.

For example:

```
SmartPointer<Server> server =new Server();
server.connect();
...
server.disconnect();
```

You do not need to delete the memory for pointer server. SmartPointer handles all memory management for you.

Example Application

The sample code shown below illustrates how to use the C++ ASCII version API.

```
try{

   //Create Server
   SmartPointer<Server> server =new Server();

   //Set server connection properties
   server->setConnectionProperty(Server::HOST, "localhost");
   server->setConnectionProperty(Server::PORT, "10119");
   server->setConnectionProperty(Server::CONNECTION_TYPE , "SOCKET");
   server->setConnectionProperty(Server::ACCOUNT_ID, "guest");
   server->setConnectionProperty(Server::ACCOUNT_PASSWORD, "");

   //Connect to server
   server->connect();

   //Get Service From Server
   SmartPointer<Service> service = server- >getService("ValidateAddress");


   //Create Input Message
   SmartPointer<Message> request = new Message();

   //Fill DataTable in the input message
   SmartPointer<DataTable> dataTable = request->getDataTable();
    SmartPointer<DataRow> row1 = dataTable->newRow();
    row1->set("AddressLine1", "4200 Parliament Place") ;
    row1->set("City", "Lanham");
    row1->set("StateProvince", "Maryland");
    dataTable->addRow(row1);

    SmartPointer<DataRow> row2 = dataTable->newRow();
    row2->set("AddressLine1", "100 Congress");
    row2->set("City", "Austin");
```

```
    row2->set("StateProvince", "Texas");
    dataTable->addRow(row2);

   //Set"option" Properties to the Input Message
    request->putOption("OutputCasing", "M");
    request->putOption("OutputRecordType", "A");

   //Process Input Message, return output Message
    SmartPointer<Message> reply = service->process(request);

    //Disconnect from server
   server->disconnect();

   //Get the result from the resonse message
   SmartPointer<DataTable> returnDataTable = reply->getDataTable();

    vector<string> columnName = returnDataTable- >getColumnNames();
      vector< SmartPointer<DataRow> >::iterator iter =
returnDataTable->iterator();

     for (int i=0; i< returnDataTable->getRowCount(); i++, iter++)
   {
   SmartPointer<DataRow> dataRow = *iter;

   for (int col = 0; col < returnDataTable->getColumnCount(); col++)
   {
   const char* value = dataRow->get(columnName[col].c_str());
   cout << value << "\n";
  }
   }
   }catch(Exception e)
   {
   cout << "Error Occurred, " << e.getErrorMessage();
   }
```

The sample code shown below illustrates how to use the C++ Unicode version API.

```
try{
 //Create Server
 SmartPointer<Server> server =new Server();

 //Set server connection properties
 server->setConnectionProperty(Server::HOST,"localhost");
 server->setConnectionProperty(Server::PORT, "10119");
 server->setConnectionProperty(Server::CONNECTION_TYPE , "SOCKET");
 server->setConnectionProperty(Server::ACCOUNT_ID, "guest");
 server->setConnectionProperty(Server::ACCOUNT_PASSWORD, "");

 //Connect to server
 server->connect();

 //Get Service From Server
 //NOTE: ValidateAddress does not support unicode, but supports
 //characters in Canadian address and International address data files.
 SmartPointer<Service> service = server- >getService("ValidateAddress");

 //Create Input Message
 SmartPointer<Message> request = new Message();

 //Fill DataTable in the input message
 SmartPointer<DataTable> dataTable = request->getDataTable();
 dataTable->addColumn("AddressLine1");
 dataTable->addColumn("City");
 dataTable->addColumn("PostalCode");
 dataTable->addColumn("Country");

 SmartPointer<DataRow> row1 = dataTable->newRow();

 UnicodeString address1 = "74, Rue Octave Bénard";
```

```
 row1->set( 0 , address1);
 UnicodeString city1 = "Etang-Salé-les-Bains";
 row1->set( 1 , city1);
 UnicodeString postalCode1 = "97427";
 row1->set( 2 , postalCode1);
 UnicodeString country1 = "Reunion Island";
 row1->set( 3 , country1);

 dataTable->addRow(row1);

 SmartPointer<DataRow> row2 = dataTable->newRow();
 UnicodeString address2 = "Final Av. Panteón Foro Libertador";
 row2->set( 0 , address2);
 UnicodeString city2 = "Caracas";
 row2->set( 1 , city2);
 UnicodeString postalCode2 = "1010";
 row2->set( 2 , postalCode2);
 UnicodeString country2 = "Venezuela";
 row2->set( 3 , country2);

 dataTable->addRow(row2);

 //Set"option" Properties to the Input Message
 request->putOption("OutputCasing", "M");
 request->putOption("OutputRecordType", "A");

 //Process Input Message, return output Message
 SmartPointer<Message> reply = service->process(request);

 //Disconnect from server
 server->disconnect();

 //Get the result from the resonse message
 SmartPointer<DataTable> returnDataTable = reply->getDataTable();

 vector<UnicodeString> columnName = returnDataTable- >getColumnNames();

 vector< SmartPointer<DataRow> >::iterator iter = returnDataTable-
>iterator();

 for (int i=0; i< returnDataTable->getRowCount(); i++, iter++)
 {
 SmartPointer<DataRow> dataRow = *iter;

 for (int col = 0; col < returnDataTable->getColumnCount(); col++)
 {
 UnicodeString value = dataRow->get(columnName[col]);
 wcout <<value.getTerminatedBuffer() <<"\n"; }
 }

 }catch(Exception e)
 {
 UnicodeString error = e.getErrorMessage() ;

 wcout << error.getTerminatedBuffer();
 }
```

Server

The Server class is used to connect to the server, disconnect from the server, and get the service from the server. The following table summarizes the functions each method performs in the Server class.

Constructors

Constructors for the Server class are as follows:

- Server()

Destructor

The Destructor for the Server class is:

- ~Server()

Connect

Reads the properties to determine the configuration settings and makes a connection to the server. You can connect via HTTP, HTTPS, or SOCKET.

Note: C++ uses the HTTP, HTTPS, or SOCKET server connection protocol. HTTP and HTTPS logically establish a client connection but do not actually connect to the server until a GetService or Process method is invoked. The SOCKET protocol establishes a connection to the server when Connect is invoked.

Syntax

```
void connect()
```

Parameters

None.

Results

Establishes client connection to the server.

Example

```
//Create Server
SmartPointer<Server> server =new Server();

//Set server connection properties
server->setConnectionProperty(Server::HOST,"localhost");
server->setConnectionProperty(Server::PORT, "10119");
server->setConnectionProperty(Server::CONNECTION_TYPE , "SOCKET");
server->setConnectionProperty(Server::ACCOUNT_ID, "guest");
server->setConnectionProperty(Server::ACCOUNT_PASSWORD, "");

//Connect to server
server->connect();
```

Disconnect

Disconnects from the server.

Syntax

```
void disconnect()
```

Parameters

None.

Results

Client is disconnected from the server.

Example

```
SmartPointer<Server> server =new Server()
server->connect();
...
server->disconnect();
```

SetConnectionProperty

Establishes the server connection configuration properties, such as host name and length of timeout.

Syntax

ASCII version:

```
void setConnectionProperty(const char* name, const char* value)
```

Unicode version:

```
void setConnectionProperty(const UnicodeString name, const UnicodeString
value)
```

Parameters

- Name — the name of the connection property, such as HOST
- Value — the value for the name of the connection property, such as "www.myhost.com"

Results

The configuration properties for connection to the server are set.

Example

ASCII Version

```
SmartPointer<Server> server =new Server()
server->setConnectionProperty(Server::HOST,"localhost");
server->setConnectionProperty(Server::PORT, "8080");
```

Unicode Version

Same as ASCII, or:

```
SmartPointer<Server> server =new Server()
UnicodeString host="localhost";// Or input unicode string
server->setConnectionProperty(Server::HOST, host);
```

GetService

Gets the service from the server.

Note: See the Component Reference section of this guide for a list of servies that may be available to you.

Syntax

ASCII Version:

```
SmartPointer<Service> getService(const char* serviceName)
```

Unicode Version:

```
SmartPointer<Service> getService(const UnicodeString serviceName)
```

Parameters

- Name of service

Results

Returns the specific service.

Example

ASCII Version

```
// Get Service From Server
SmartPointer<Service> service = server- >getService("ValidateAddress");
```

Unicode Version

Same as ASCII, or:

```
// Get Service From Server
UnicodeString serviceName="ValidateAddress";// Or input unicode string
SmartPointer<Service> service = server->getService(serviceName);
```

Service

The Service class is used to process the message (i.e., send the message to the server and receive a response from the server).

Process

Processes the input message and returns the response message.

Syntax

```
SmartPointer<Message> process(Message* message)
```

Parameters

- Input message

Results

Returns the response message.

Example

```
SmartPointer<Message> reply = service->process(request);
```

Message

The Message class sends your input data and receives your output data from the service. The properties for Message include context entities, such as account ID, account password, service name, and service method; option entities, which are the Service-specific runtime options; and error entities, which are the error class, error message and error stacktrace.

Constructors

Constructors for the Message class are as follows:

- Message()

For example:

```
Message *request = new Message():
```

- Message(const Message&)

For example:

```
Message* request = new Message();
Message anotherMessage = request;
Message message(anotherMessage);
```

Destructor

The Destructor for the Message class is:

- ~Message();

The following table summarizes the functions each method performs in the Message class.

Table 16: Message Methods Summary

Method	Function
getContext	Gets the value of the context entity identified by the name in the context session of the message.
getContext	Gets the Map that contains all of the context entries.
putContext	Sets the value of the context entity identified by the name in the context session of the message. If there is an existing value present for the entity identified by the name, it is replaced.

Method	Function
putContext	Adds the new context properties to the current context properties.
setContext	Overwrites the current context properties with the new context properties.
getOption	Gets the value of the option entity identified by the name in the option session of the message.
getOptions	Gets the Map that contains all of the option entries.
putOption	Sets the value of the option entity identified by the name in the option session of the message. If there is an existing value present for the entity identified by the name, it is replaced.
putOptions	Adds the new option properties to the current option properties.
setOptions	Overwrites the current option properties with the new option properties.
getError	Gets the error message.
getDataTable	Gets the DataTable from the message.

GetContext

Gets the value of the context entity identified by the name in the context session of the message.

Syntax

ASCII Version

```
const char* getContext(const char* name)
```

Unicode Version

```
const UnicodeString getContext(const UnicodeString name)
```

Parameters

- The name whose associated value is to be returned

Results

Returns the value for the name in the context entity. If the name does not exist, the method returns empty string.

Example

ASCII Version

```
const char* value= msg->getContext(Server::ACCOUNT_ID);
```

Unicode Version

Same as ASCII or:

```
UnicodeString name= Server::ACCOUNT_ID;// Or input unicode string
UnicodeString value= msg->getContext(name);
```

GetContext

Gets the Map that contains all of the context entries.

Syntax

ASCII Version

```
map<string , string> getContext()
```

Unicode Version

```
map< UnicodeString, UnicodeString > getContext()
```

Parameters

None.

Results

Returns the map that contains all of the context entries.

Example

ASCII Version

```
map<string , string> context = message->getContext();
```

Unicode Version

```
map< UnicodeString, UnicodeString > context = message- >getContext();
```

PutContext

Sets the value for given name in the context properties. If there is an existing value present for the entity identified by the name, it is replaced. Context properties include the following constants: account ID, account password, service name, service key, and request ID.

Syntax

ASCII Version

```
void putContext(const char* name, const char* value)
```

Unicode Version

```
void putContext(const UnicodeString name, const UnicodeString value)
```

Parameters

- Name with which the specified value is to be associated.
- Value to be associated with the specified name

Example

ASCII Version

```
message->putContext(Message.CONTEXT_ACCOUNT_ID, "user1");
```

Unicode Version

Same as ASCII or:

```
UnicodeString account="user1" ;// Or input unicode string
message->putContext(Message.CONTEXT_ACCOUNT_ID, account);
```

PutContext

Adds the new context properties to the current context properties.

Syntax

ASCII Version

```
void putContext(map<string , string> context)
```

Unicode Version

```
void putContext(map< UnicodeString, UnicodeString > context)
```

Parameters

- The new context map to be added to the current context map

Example

ASCII Version

```
map<string , string> context ;
//more code
message->putContext(context);
```

Unicode Version

```
map< UnicodeString, UnicodeString > context ;
//more code
message->putContext(context);
```

SetContext

Overwrites the current context properties with the new context properties.

Syntax

ASCII Version

```
void setContext(map<string , string> context)
```

Unicode Version

```
void setContext(map< UnicodeString, UnicodeString > context)
```

Parameters

- The new context map to be used to replace the current context map

Example

ASCII Version

```
map<string , string> context ;
//more code
message->setContext(context);
```

Unicode Version

```
map< UnicodeString, UnicodeString > context ;
//more code
message->setContext(context);
```

GetOption

Gets the value of the option entity identified by name in the option section of the message. Option entities include the service-specific runtime options, such as output casing, output data format, and so on.

Syntax

ASCII Version

```
const char* getOption(const char* name)
```

Unicode Version

```
const UnicodeString getOption(const UnicodeString name)
```

Parameters

- The name whose associated value is to be returned

Results

Returns the value for the name in the context entity. If the name does not exist, the method returns empty string.

Example

ASCII Version

```
const char* value = message->getOption("OutputCasing");
```

Unicode Version

Same as ASCII or:

```
UnicodeString option="OutputCasing"; // Or input unicode string
UnicodeString value= message->getOption(option);
```

GetOptions

Gets the map that contains all of the option entries.

Syntax

ASCII Version

```
map<string , string> getOptions()
```

Unicode Version

```
map< UnicodeString, UnicodeString > getOptions()
```

Parameters

None.

Results

Returns the map that contains all of the option entries.

Example

ASCII Version

```
const char* value = message->getOption("OutputCasing");
```

Unicode Version

```
UnicodeString option="OutputCasing"; //or input Unicode string
UnicodeString value= message->getOption(option);
```

PutOption

Sets the value for given name in the option properties. If there is an existing value present for the entity identified by the name, it is replaced. Option properties are the service-specific run-time options.

Syntax

ASCII Version

```
void putOption(const char* name, const char* value)
```

Unicode Version

```
void putOption(const UnicodeString name, const UnicodeString value)
```

Parameters

- Name with which the specified value is to be associated
- Value to be associated with the specified name

Example

ASCII Version

```
message->putOption("OutputCasing", "M");
```

Unicode Version

Same as ASCII or:

```
UnicodeString option="M"; // Or input unicode string
message->putOption("OutputCasing", option);
```

PutOptions

Adds the new option properties to the current option properties.

Syntax

ASCII Version

```
void putOptions(map<string , string> options)
```

Unicode Version

```
void putOptions(map< UnicodeString, UnicodeString > options)
```

Parameters

- The new option map to be added to the current option properties

Example

ASCII Version

```
map<string , string> options ;
//more code
message->putOptions(options);
```

Unicode Version

```
map< UnicodeString, UnicodeString > options ;
//more code
message->putOptions(options);
```

SetOptions

Overwrites the current option properties with the new option properties.

Syntax

ASCII Version

```
void setOptions(map<string , string> options)
```

Unicode Version

```
void setOptions(map< UnicodeString, UnicodeString > options)
```

Parameters

- The new option map to be used to replace the current option map

Example

ASCII Version

```
map<string , string> options ;
//more code
message->setOptions(options);
```

Unicode Version

```
map< UnicodeString, UnicodeString > options ;
//more code
message->setOptions(options);
```

GetError

Gets the error message from the message.

Syntax

ASCII Version

```
string getError()
```

Unicode Version

```
UnicodeString getError()
```

Parameters

None.

Results

Returns the error message in message

Example

ASCII Version

```
String error = message->getError();
```

Unicode Version

```
UnicodeString error = message->getError();
```

GetDataTable

Gets the DataTable in the message.

Syntax

```
SmartPointer<DataTable> getDataTable()
```

Parameters

None.

Example

```
SmartPointer<DataTable> dataTable
= message->getDataTable();
```

DataTable

DataTable contains the records for the input and output data.

Constructors

Constructors for the DataTable class are as follows:

- DataTable()

For example:

```
DataTable* dataTable = new DataTable()
```

Destructor

The Destructor for the DataTable class is:

- ~DataTable();

The following table summarizes the functions each method performs in the DataTable class.

Table 17: DataTable Methods Summary

Method	Function
addColumn	Adds the new column.
getColumnNames	Gets all the column names.
getColumnIndex	Gets the corresponding column index.
getColumnCount	Gets the number of columns.
clear	Clears the data in DataTable.
iterator	An iterator that contains all DataRows in the DataTable.
addRow	Adds a DataRow to the DataTable.
newRow	Creates a new DataRow in the DataTable.
getRowCount	Gets the number of the DataRows in this DataTable.
merge	Merges the given DataTable and the current DataTable.

AddColumn

Adds the new column.

Syntax

ASCII Version

```
int addColumn(const char* columnName)
```

Unicode Version

```
int addColumn(const UnicodeString columnName)
```

Parameters

- Column name

Results

- Returns the index of column

Exceptions

- Blank column name
- Duplicate column name

Example

ASCII Version

```
SmartPointer<DataTable> dataTable = message.getDataTable();
dataTable->addColumn("Address");
dataTable->addColumn("City");
```

Unicode Version

Same as ASCII or:

```
SmartPointer<DataTable> dataTable = message.getDataTable();
UnicodeString columnName="Address"; // Or input unicode string
dataTable->addColumn(columnName);
```

GetColumnNames

Gets all the column names.

Syntax

ASCII Version

```
vector<string> getColumnNames();
```

Unicode Version

```
vector<UnicodeString> getColumnNames();
```

Parameters

None.

Results

Returns the vector of column names

Example

ASCII Version

```
vector<string> columnNames = dataTable->getColumnNames();
```

Unicode Version

```
vector<UnicodeString> columnNames = dataTable->getColumnNames();
```

GetColumnIndex

Gets the corresponding column index.

Syntax

ASCII Version

```
int getColumnIndex(const char* columnName)
```

Unicode Version

```
int getColumnIndex(const UnicodeString columnName)
```

Parameter

- Column name

Results

Returns the corresponding column index.

Example

ASCII Version

```
int columnIndex = dataTable->getColumnIndex ("City");
```

Unicode Version

Same as ASCII or:

```
UnicodeString columnName="City"; // Or input unicode string
int columnIndex = dataTable->getColumnIndex (columnName);
```

GetColumnCount

Gets the number of columns.

Syntax

```
int getColumnCount()
```

Parameter

None.

Results

Returns the number of columns.

Example

```
int columnCount = dataTable->getColumnCount ();
```

Clear

Clears the data in DataTable.

Syntax

```
void clear()
```

Parameters

None.

Example

```
dataTable->clear();
```

Iterator

An iterator that contains all DataRows in the DataTable.

Syntax

```
vector< SmartPointer<DataRow> >::iterator iterator()
```

Parameters

None.

Results

Returns an iterator that contains all DataRows in the DataTable.

Example

```
vector<string> columnName
= returnDataTable->getColumnNames();

vector< SmartPointer<DataRow> >::iterator theIterator
= returnDataTable->iterator();
```

```
for (int i=0; i< returnDataTable->getRowCount();
i++, theIterator++)
{
SmartPointer<DataRow> dataRow = *theIterator;

for (int col = 0;
col < returnDataTable->getColumnCount(); col++)
 {
 const char* value = dataRow->get(columnName[col].c_str());
 }
 }
```

AddRow

Adds a DataRow to the DataTable.

Syntax

```
void addRow( SmartPointer<DataRow> dataRow)
```

Parameters

- DataRow to be added to the DataTable

Example

```
SmartPointer<DataRow> newRow = dataTable->newRow();
 newRow->set( 0 , "10535 Boyer");
 newRow->set( 1 , "Austin");
 newRow->set( 2 , "Texas");
 dataTable->addRow(newRow);
```

NewRow

Creates a new DataRow in the DataTable.

Syntax

```
SmartPointer<DataRow> newRow()
```

Results

Returns the new created DataRow

Example

```
SmartPointer<DataRow> newRow = dataTable->newRow();
 newRow->set( 0 , "10535 Boyer");
 newRow->set( 1 , "Austin");
 newRow->set( 2 , "Texas");
 dataTable->addRow(newRow);
```

GetRowCount

Gets the number of the DataRows in this DataTable.

Syntax

```
int getRowCount()
```

Results

Returns the number of the DataRows in this DataTable.

Example

```
int rowCount = dataTable->getRowCount();
```

Merge

Merges the given DataTable and the current DataTable.

Syntax

```
void merge(DataTable* other)
```

Parameters

- Other DataTable to be merged with the current DataTable

Example

```
DataTable* otherDataTable = new DataTable();
dataTable->merge(otherDataTable);
```

DataRow

DataRow contains the record for the input and output data.

Constructor

Constructors for the DataRow class are as follows:

- DataRow ()

For example:

```
DataRow * dataRow = new DataRow();
```

- DataRow(const DataRow&)

For example:

```
DataRow* dataRow = new DataRow();
DataRow anotheDataRow = dataRow;
DataRow newDataRow(anotheDataRow);
```

Destructor

The Destructor for the DataRow class is:

- ~ DataRow();

The following table summarizes the functions each method performs in the DataRow class.

Table 18: DataRow Methods Summary

Method	Function
getColumnNames	Gets all the column names.
getColumnIndex	Gets the corresponding column index.
getColumnCount	Gets the number of columns.
get	Gets the value from the fields array by the column index in this DataRow.
get	Gets the value from the fields array by the column name in this DataRow.
merge	Merges the given DataTable and the current DataTable.
set	Sets the value for the corresponding column name for the DataRow. If the value for the name exists, the old value is replaced.
set	Sets the value for the corresponding column index for the DataRow. If the value for the name exists, the old value is replaced.
addChild	Adds a new DataRow to the named parent/child relationship. If the named relationship exists, the supplied DataRow will be appended to the existing DataRow Collection, otherwise a new Collection will be created with the supplied DataRow as its only element.
getChildren	Retrieves the child rows from a named relationship.
listChildNames	Retrieves all of the names of the named parent/child relationships.
setChildren	Sets the rows of a supplied, named parent/child relationship. If rows previously existed under this name, they will be returned to the caller.

GetColumnNames

Gets all the column names.

Syntax

ASCII Version

```
vector<string> getColumnNames()
```

Unicode Version

```
vector<UnicodeString> getColumnNames()
```

Parameters

None.

Results

Returns the vector of column names

Example

ASCII Version

```
vector<string> columnNames = dataRow->getColumnNames();
```

Unicode Version

```
vector<UnicodeString> columnNames = dataRow->getColumnNames();
```

GetColumnIndex

Gets the corresponding column index.

Syntax

ASCII Version

```
int getColumnIndex(const char* columnName)
```

Unicode Version

```
int getColumnIndex(const UnicodeString columnName)
```

Parameter

- Column name

Results

Returns the corresponding column index.

Example

ASCII Version

```
int columnIndex = dataRow->getColumnIndex ("City");
```

Unicode Version

Same as ASCII or:

```
UnicodeString columnName="City"; // Or input unicode string
int columnIndex = dataRow->getColumnIndex (columnName);
```

GetColumnCount

Gets the number of columns.

Syntax

```
int getColumnCount()
```

Parameter

None.

Results

Returns the number of columns.

Example

```
int columnCount = dataRow->getColumnCount ();
```

Get

Gets the value from the fields array by the column index in this DataRow.

Syntax

ASCII Version

```
const char* get(int index)
```

Unicode Version

```
const UnicodeString get(int index)
```

Parameters

- Index with which the specified value is to be associated

Results

Returns the value for the column index in the DataRow, returns empty string if the index in invalid.

Example

ASCII Version

```
const char* value = dataRow->get(1);
```

Unicode Version

```
const UnicodeString value = dataRow->get(1);
```

Get

Gets the value from the fields array by the column name in this DataRow

Syntax

ASCII Version

```
const char* get(const char* columnName)
```

Unicode Version

```
const UnicodeString get(const UnicodeString columnName)
```

Parameters

- Name with which the specified value is to be associated

Results

Returns the value for the column name in the DataRow, returns empty string if the column name does not exist.

Example

ASCII Version

```
const char* value = dataRow->get("City");
```

Unicode Version

Same as ASCII, or:

```
UnicodeString columnName="City"; // Or input unicode string
const UnicodeString value = dataRow->get(columnName);
```

Merge

Merges the given DataRow and the current DataRow.

Syntax

```
void merge(DataRow* other)
```

Parameters

- Other DataRow to be merged with the current DataRow

Example

```
DataRow* otherDataRow = new DataRow();
DataRow->merge(otherDataRow);
```

Set

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

ASCII Version

```
void set(const char* columnName, const char* value)
```

Unicode Version

```
void set(const UnicodeString columnName, const UnicodeString value)
```

Parameters

- Name with which the specified value is to be associated
- Value to be associated with the specified name

Exceptions

- Blank column name
- Duplicate column name

Example

ASCII Version

```
SmartPointer<DataRow> newRow = dataTable->newRow();
newRow->set( "AddressLine1" , "10535 Boyer");
newRow->set( "City" , "Austin");
newRow->set( "State" , "Texas");
```

Unicode Version

Same as ASCII or:

```
SmartPointer<DataRow> newRow = dataTable->newRow();
UnicodeString address="10535 Boyer"; // Or input unicode string
newRow->set( "AddressLine1" , address);
```

Set

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

ASCII Version

```
void set(int index, const char* value)
```

Unicode Version

```
void set(int index, const UnicodeString value)
```

Parameters

- Column index with which the specified value is to be associated
- Value to be associated with the specified name

Exceptions

- The column index is invalid.

Example

ASCII Version

```
SmartPointer<DataRow> newRow = dataTable->newRow();
newRow->set( 0 , "10535 Boyer");
newRow->set( 1 , "Austin");
newRow->set( 2 , "Texas");
```

Unicode Version

Same as ASCII or:

```
SmartPointer<DataRow> newRow = dataTable->newRow();
UnicodeString address="10535 Boyer"; // Or input unicode string
newRow->set( 0 , address);
```

AddChild

Adds a new DataRow to the named parent/child relationship. If the named relationship exists, the supplied DataRow will be appended to the existing DataRow collection. Otherwise, a new collection will be created with the supplied DataRow as its only element.

Syntax

ASCII Version

```
void addChild(const char* childName, SmartPointer<DataRow> childDataRow)
```

Unicode Version

```
void addChild(const UnicodeString childName, SmartPointer<DataRow>
childDataRow)
```

Parameters

- The name of the parent/child relationship (for example, "Flood Plain Data," "References, " "Used By")
- The DataRow to be added to the relationship.

Example

```
SmartPointer<DataRow> childDataRow =new DataRow();
childDataRow ->set("Address", "100 Congress");
childDataRow ->set("City", "Austin");
SmartPointer<DataRow> dataRow =new DataRow();
dataRow->addChild("child1", childDataRow );
```

GetChildren

Retrieves the child rows from a named relationship.

Syntax

ASCII Version

```
list< SmartPointer<DataRow> > getChildren(const char* childName)
```

Unicode Version

```
list< SmartPointer<DataRow> > getChildren(const UnicodeString childName)
```

Parameters

- The name of the parent/child relationship, e.g. "Flood Plain Data", "References", "Used By", etc.

Results

Returns the child rows from the named relationship.

Example

```
list< SmartPointer<DataRow> > rowsChild2= dataRow- >getChildren("child2");
```

ListChildNames

Retrieves all of the names of the named parent/child relationships.

Syntax

ASCII Version

```
list<string> listChildNames()
```

Unicode Version

```
list<UnicodeString> listChildNames()
```

Results

Returns the set of the names of the named parent/child relationships.

Example

```
list<G1CLIENT_STRING> names = dataRow->listChildNames();
```

SetChildren

Sets the rows of a supplied, named parent/child relationship. If rows previously existed under this name, they will be returned to the caller.

Syntax

ASCII Version

```
list< SmartPointer<DataRow> > setChildren(const char* childName, list<
SmartPointer<DataRow> > dataRows)
```

Unicode Version

```
list< SmartPointer<DataRow> > setChildren(const UnicodeString childName,
list< SmartPointer<DataRow> > dataRows)
```

Results

Returns the set of the names of the named parent/child relationships.

Example

```
SmartPointer<DataRow> dataRow1=new DataRow();
dataRow1->set("Address", "100 Congress");
dataRow1->set("City", "Austin");
SmartPointer<DataRow> dataRow2=new DataRow();
dataRow2->set("Address", "200 Congress");
dataRow2->set("City", "Austin");
list< SmartPointer<DataRow> > rows ;
rows.push_back(dataRow1);
rows.push_back(dataRow2);
list< SmartPointer<DataRow> > rowsNewChildren = dataRowSpt-
>setChildren("child1", rows);
```

4

The COM API

In this section:

- Introduction 100
- Server 103
- Service 105
- Message 106
- DataTable 111
- DataRow 116
- Map 122

Introduction

The term Component Object Model (COM) refers to an open architecture for cross-platform development of client/server applications based on object-oriented technology. COM is a way of creating reusable software components. Clients have access to an object through interfaces implemented on the object. In other words, objects are the communication medium between client and server. The Component Object Model provides a flexible way for building distributed object-oriented systems. COM objects are language-independent, can be shipped in binary form, can be upgraded without requiring changes to the existing integrated code, and can be transparently relocated on a network. Because of these qualities, COM objects are extremely flexible and can be adapted to add specific functionality to almost any Windows-based client-server system.

Note: The examples shown in this chapter are written in Visual Basic.

The Spectrum™ Technology Platform COM API consists of the following interfaces:

- Server
- Service
- Message
- DataTable
- DataRow
- Map

These interfaces are described in detail in the following sections of this chapter.

Constants

The COM API uses two sets of constants. The first set is for the Server object, described in the table below.

Table 19: Constants for the Server Component

Constant Name	Description/Default	Example
SERVER.HOST	String for server host name. Default is "localhost".	65.89.200.89
SERVER.PORT	String for server port. Default is "8080".	10119
SERVER.ACCOUNT_ID	String for server account ID. No default value.	user1
SERVER.ACCOUNT_PASSWORD	String for server account password. No default value.	user1
SERVER.CONNECTION_TIMEOUT	String for server connection timeout, in milliseconds. Default is "5000".	50000
SERVER.CONNECTION_TYPE	String for server connection type. Currently only supports HTTP, HTTPS, or SOCKET. Default is "HTTP".	HTTP(S)
SERVER.PROXY_HOST	String for proxy server host name. No default value.	192.168.1.77
SERVER.PROXY_PORT	String for proxy server port. No default value.	8080

Constant Name	Description/Default	Example
SERVER.PROXY_USER	String for proxy server account ID. No default value.	user1
SERVER.PROXY_PASSWORD	String for proxy server account password. No default value.	user1

The second set of constants is for the Message component.

Table 20: Constants for the Message Component

Constant Name	Description	Example
MESSAGE.CONTEXT_ACCOUNT_ID	String for message context account ID.	user1
MESSAGE.CONTEXT_ACCOUNT_PASSWORD	String for message context account password.	user1
MESSAGE.CONTEXT_SERVICE_NAME	String for message context service name.	echoservice

Error Messages

The COM API uses the following error messages:

- Error Messages for Connection:
  - "Connection type not supported"
  - "Client timeout"
- Error Messages for creating DataTable:
  - "Blank column name"
  - "Duplicated column name"
  - "The column index is invalid"
- Error Messages for Message Packaging Exception:
  - "Input Message is null"
  - "Failed to connect to Server"
  - "Failed to disconnect to Server"
  - "Failed to open Http Connection"
  - "Failed to get Service"
  - "Failed to package the message using Serializer and Encoding"

For example:

```
On Error GoTo ErrorHandler
  Dim server As New G1CLIENTLib.server
  server.setConnectionProperty server.HOST, "localhost"
  server.setConnectionProperty server.Port, "8080"
  'Making connection to the server
  server.Connect
  ...
```

```
  Exit Sub
  ErrorHandler:
  MsgBox Err.Description
```

Example Application

The sample code shown below illustrates how to use the COM API.

```
On Error GoTo ErrorHandler

 Dim server As New G1CLIENTLib.server
 Dim service As G1CLIENTLib.service
 Dim requestMsg As New G1CLIENTLib.Message
 Dim replyMsg As G1CLIENTLib.Message
 Dim dataTable As G1CLIENTLib.dataTable
 Dim newRow As G1CLIENTLib.dataRow
 Dim returnDataTable As G1CLIENTLib.dataTable
 Dim row As G1CLIENTLib.DataRow
 Dim sColumnNames() As String
 Dim sColumnName As String
 Dim sFieldValue As String
 Dim rows() As Variant
 Dim nRow As Integer
 Dim nColumn As Integer
 'Set server connection properties
 server.setConnectionProperty server.HOST, "localhost"
 server.setConnectionProperty server.Port, "10119"
 server.setConnectionProperty server.CONNECTION_TYPE, "SOCKET"
 server.setConnectionProperty server.ACCOUNT_ID, "guest"
 server.setConnectionProperty server.ACCOUNT_PASSWORD, ""

 'Connect to server
 server.Connect

 'Get the service from the server
 Set service = server.getService("ValidateAddress")

 'Fill DataTable in the input message
 Set dataTable = requestMsg.getDataTable
 dataTable.addColumn ("AddressLine1")
 dataTable.addColumn ("City")
 dataTable.addColumn ("StateProvince")

 Set newRow = dataTable.newRow
 newRow.setByIndex 0, "10535 Boyer"
 newRow.setByIndex 1, "Austin"
 newRow.setByIndex 2, "Texas"
 dataTable.addRow newRow

 'Set"option" Properties to the Input Message
 requestMsg.putOption "OutputCasing", "M"
 requestMsg.putOption "OutputRecordType", "A"

 'Process Input Message, return output Message
 Set replyMsg = service.process(requestMsg)

 'Disconnect from the server
 server.disconnect

 'Get the result from the resonse message
 Set returnDataTable = replyMsg.getDataTable
 ReDim rows(returnDataTable.getRowCount) As Variant

 rows = returnDataTable.iterator

 ReDim sColumnNames(returnDataTable.getColumnCount) As String
 sColumnNames = returnDataTable.getColumnNames
```

```
 For nRow = 0 To returnDataTable.getRowCount - 1
Set row = rows(nRow)

For nColumn = 0 To row.getColumnCount - 1
  sColumnName = sColumnNames(nColumn)
  sFieldValue = row.getByName(sColumnName)
Next

 Next

  Exit Sub

  ErrorHandler:

MsgBox Err.Description
```

Server

Use the Server object to connect to the server, disconnect from the server, and get the service from the server. The following table summarizes the functions each method performs in the Server object.

Connect

Connects to the server. You can connect via HTTP or SOCKET.

Note: COM uses the HTTP, HTTPS, or SOCKET server connection protocol. HTTP and HTTPS logically establish a client connection but do not actually connect to the server until a GetService or Process method is invoked. The SOCKET protocol establishes a connection to the server when Connect is invoked.

Syntax

```
Sub connect()
```

Parameters

None.

Results

None.

Exception

Connection type not supported.

Example

```
Dim server As New G1CLIENTLib.server
server.connect
```

Disconnect

Disconnects from the server.

Syntax

```
Sub disconnect()
```

Parameters

None.

Results

None.

Examples

```
Dim server As New G1CLIENTLib.server
server.disconnect
```

GetService

Gets the service (such as ValidateAddress) from the server.

Note: See the Component Reference section of this guide for a list of services that may be available to you.

Syntax

```
Function getService(serviceName As String) As Service
```

Parameters

- serviceName - the name of the service which the client requires

Results

The requested service or NULL if the service does not exist.

Exceptions

- ERROR_FAIL_TO_GET_SERVICE — if there is no connection to the server.

Example

```
Dim server As New G1CLIENTLib.server
Dim service As G1CLIENTLib.service
...
'get the service from the server
Set service = server.getService("ValidateAddress")
```

SetConnectionProperty

Establishes the server connection configuration properties, such as host name and length of timeout.

Syntax

```
Sub setConnectionProperty(name As String, value As String)
```

Parameters

- Name — the name of the connection property, such as HOST
- Value — the value for the name of the connection property, such as "www.myhost.com"

Results

Return codes — none.

Exceptions

- ERROR-INVALID-COLUMN_NAME — an empty or null column name.
- ERROR_INVALID_VALUE — A null value.

Example

```
set connection properties
Dim server As New G1CLIENTLib.server

server.setConnectionProperty server.HOST,"localhost"
server.setConnectionProperty server.PORT, "8080"
```

Service

Service calls the service and processes the message you are sending (in other words, it sends the input message and receives the response).

Process

Processes the input message and gets back the response message from the server.

Syntax

```
Function process(IRequest As Message) As Message
```

Parameters

- iRequest— the input message object that contains the "option" setting and the DataTable

Results

Returns the response message for the request.

Exceptions:

- ERROR_NULL_INPUT_MESSAGE — Request message is null.

Example

```
Dim service As New G1CLIENTLib.service
Dim replyMsg As G1CLIENTLib.Message
...
'Process the message and return back the response message
Set replyMsg = service.process(requestMsg)
```

Message

Messages are used to send your input data and receive your output data from the service. The properties for Message include "context entities, such as account ID, account password, service name, and service method; "option entities, which are the service-specific runtime options; and "error entities, which are the error class, error message and error stacktrace.

GetContext

Gets the value of the context entity identified by name in the context section of the message. "Context" entities include the following constants: account ID, account password, service name, and service method.

Syntax

```
Function getContext(name As String) As String
```

Parameters

- Name—the name whose associated value is to be returned

Results

String — the value of the named entity or empty string if the named entity does not exist.

Example

```
Dim msg As New G1CLIENTLib.Message
Dim accountID As String

accountID = msg.getContext(msg.CONTEXT_ACCOUNT_ID)
```

GetContextMap

Gets the Map that contains all of the context entries.

Syntax

```
Function getContextMap() As Map
```

Parameters

- None

Results

Returns the Map that contains all of the context entries.

Example

```
Dim map As G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message
Dim sKey As String
Dim sValue As String
```

```
requestMsg.putContext
 requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext
 requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

PutContext

Sets the value for the given name in the context properties. If there is an existing value present for the entity identified by the name, it is replaced. "Context" properties include the following constants: account ID, account password, service name, and service method.

Syntax

```
Sub putContext(name As String, value As String)
```

Parameters

- Name—the name with which the specified value is to be associated
- Value—value to be associated with the specific name.

Results

None.

Example

```
Dim requestMsg As New G1CLIENTLib.Message

requestMsg.putContext
 requestMsg.CONTEXT_ACOUNT_ID, "admin"
requestMsg.putContext
 requestMsg.CONTEXT_ACOUNT_PASSWORD, "admin"
```

PutContextMap

Adds the new context properties to the current context properties.

Syntax

```
Sub putContextMap(context As Map)
```

Parameters

- The new context map to be added to the current context map

Results

None.

Example

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1UBCAPICOMLib.Message

map.Insert requestMsg.CONTEXT_ACCOUNT_ID, "admin"
map.Insert requestMsg.CONTEXT_ACOUNT_PASSWORD, "admin"

requestMsg.putContextMap map
```

SetContextMap

Overwrites the current context properties with the new context properties.

Syntax

```
Sub setContextMap(context As Map)
```

Parameters

- The new context map to replace the current context map

Results

None.

Example

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1UBCAPICOMLib.Message

map.Insert requestMsg.CONTEXT_ACCOUNT_ID, "admin"
map.Insert requestMsg.CONTEXT_ACOUNT_PASSWORD, "admin"

requestMsg.setContextMap map
```

GetOption

Gets the value of the option entity identified by name in the option section of the message. Option entities include the service-specific runtime options, such as output casing, output data format, and so on.

Syntax

```
Function getOption(name As String) As String
```

Parameters

- Name — the name whose associated value is to be returned

Results

- String — the value of the named entity or empty string if the named entity does not exist.

Example

```
Dim msg As New G1CLIENTLib.Message
Dim optionValue As String
```

```
OptionValue = msg.getOption("OutputCasing")
```

GetOptions

Gets the map that contains all of the option entries.

Syntax

```
Function getOptions() As Map
```

Parameters

- None

Results

Returns the map that contains all of the option entries.

Example

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message
Dim sKey As String
Dim sValue As String

requestMsg.putOption "OutputCasing", "M"
requestMsg.putOption "OutputRecordType", "A"

Set map = requestMsg.getOptions

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

PutOption

Sets the value for the given name in the option properties. If there is an existing value present for the entity identified by the name, it is replaced. "Option" properties are the service-specific runtime options.

Syntax

```
Sub putOption(name As String, value As String)
```

Parameters

- Name—the name with which the specified value is to be associated
- Value—value to be associated with the specific name.

Example

```
Dim requestMsg As New G1CLIENTLib.Message

requestMsg.putOption "OutputCasing", "M"
requestMsg.putOption "OutputRecordType", "A"
```

PutOptions

Adds the new option properties to the current option properties.

Syntax

```
Sub putOptions(options As Map)
```

Parameters

- The new option map to be added to the current option properties

Example

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message

map.Insert "OutputCasing", "M"
map.Insert "OutputRecordType", "A"

requestMsg.putOptions map
```

SetOptions

Overwrites the current option properties with the new option properties.

Syntax

```
Sub setOptions(options As Map)
```

Parameters

- The new option map to replace the current option map

Example

```
Dim map As New G1CLIENTLib.Map
Dim requestMsg As New G1CLIENTLib.Message

map.Insert "OutputCasing", "M"
map.Insert "OutputRecordType", "A"

requestMsg.setOptions map
```

GetError

Gets the error from the error message.

Syntax

```
Function getError() As String
```

Parameters

- None

Results

Returns the error message in the message.

Example

```
Dim sErrorMessage As String
...

sErrorMessage = replyMsg.getError()
```

GetDataTable

Gets the DataTable in the message.

Syntax

```
Function getDataTable() As DataTable
```

Parameters

- None.

Example

```
Dim DataTable AS G1CLIENTLib.dataTable

Set DataTable = message.getDataTable
```

DataTable

DataTable contains the records for your input and output data. Using the methods associated with this object, you define the column names for your output and add rows to the DataTable.

AddColumn

Adds the new column to the DataTable.

Syntax

```
Function addColumn(columnName As String) As Integer
```

Parameters

- Column name

Results

Returns the index of the column

Exceptions

- Blank column name

- Duplicate column name

Example

```
Dim dataTable As G1CLIENTLib.dataTable

dataTable.addcolumn "AddressLine1"
dataTable.addcolumn "City"
```

GetColumnNames

Gets all the column names.

Syntax

```
Syntax Function getColumnNames() As String()
```

Parameters

- None

Results

Returns the array of column names.

Example

```
Dim sColumnNames() As String
Dim sColumnName As String
Dim nColumn As Integer

ReDim sColumnNames(returnDataTable.getColumnCount) As String
sColumnNames = returnDataTable.getColumnNames

For nColumn = 0 To dataRow.getColumnCount - 1
sColumnName = sColumnNames (nColumn)
Next
```

GetColumnIndex

Gets the corresponding column index.

Syntax

```
Function getColumnIndex(columnName As String) As Integer
```

Parameters

- Column name

Results

Returns the corresponding column index.

Example

```
Dim nIndex As Integer
nIndex = dataTable.getColumnIndex("AddressLine1")
```

GetColumnCount

Gets the number of columns in the DataTable.

Syntax

```
Function getColumnCount() As Integer
```

Parameters

- None

Results

Returns the number of columns.

Example

```
Dim nColumnCount As Integer
nColumnCount = dataTable.getColumnCount()
```

Clear

Clears the data in the DataTable.

Syntax

```
Sub clear()
```

Parameters

- None

Example

```
dataTable.clear()
```

Iterator

An iterator that contains all DataRows in the DataTable.

Syntax

```
Syntax Function iterator() As DataRow()
```

Parameters

- None

Results

Returns an iterator that contains all DataRows in the DataTable.

Example

```
Dim returnDataTable As G1CLIENTLib.dataTable
  Dim row As G1CLIENTLib.DataRow
  Dim sColumnName As String
  Dim sFieldValue As String
  Dim rows() As Variant
  Dim nRow As Integer
  Dim nColumn As Integer

  'Get the result from the resonse message
  Set returnDataTable = replyMsg.getDataTable
  ReDim rows(returnDataTable.getRowCount) As Variant

  rows = returnDataTable.iterator

  For nRow = 0 To returnDataTable.getRowCount - 1
 Set row = rows(nRow)

 For nColumn = 0 To row.getColumnCount - 1
sColumnName = row.getColumnNames(nColumn)
sFieldValue = row.getByName(sColumnName)
 Next

 Next
```

AddRow

Adds a DataRow to the DataTable.

Syntax

```
Sub addRow(DataRow As DataRow)
```

Parameters

- DataRow to be added to the DataTable

Results

None.

Example

```
Dim dataTable As G1CLIENTLib.dataTable
Dim newRow As G1CLIENTlib.DataRow

Set dataTable=requestMsg.getDataTable
dataTable.addColumn("AddressLine1")
dataTable.addColumn("City")
dataTable.addColumn("State")
Set newRow=dataTable.newRow
newRow.setByIndex 0, "10535 Boyer"
newRow.setByIndex 1, "Austin"
newRow.setByIndex 2, "Texas"
dataTable.addRow newRow
```

NewRow

Creates a new DataRow in the DataTable.

Syntax

```
Function newRow() As DataRow
```

Parameters

- None

Results

Returns the newly created DataRow

Example

```
Dim dataTable As G1CLIENTLib.dataTable
Dim newRow As G1CLIENTlib.DataRow

Set dataTable=requestMsg.getDataTable

Set newRow=dataTable.newRow
newRow.setByName "AddressLine1","10535 Boyer"
newRow.setByName "City", "Austin"
newRow.setByName "State", "Texas"
dataTable.addRow newRow
```

GetRowCount

Gets the number of DataRows in the DataTable.

Syntax

```
Function getRowCount() As Integer
```

Parameters

- None

Results

Returns the number of DataRows in the DataTable.

Example

```
Dim nRowCount As Integer
nRowCount = dataTable.getRowCount
```

Merge

Merges the given DataTable and the current DataTable.

Syntax

```
Sub merge(other As DataTable)
```

Parameters

- The other DataTable to be merged with the current DataTable

Results

None.

Example

```
Dim otherDataTable As New G1CLIENTlib.dataTable
...
dataTable.merge(otherDataTable)
```

DataRow

DataRow contains the individual records for your input and output data. Using the methods associated with this class, you define the column names for your output and add records to the DataTable.

GetColumnNames

Gets all the column names.

Syntax

```
Function getColumnNames() As String()
```

Parameters

- None

Results

Returns the array of column names.

Example

```
Dim sColumnNames() As String
Dim sColumnName As String
Dim nColumn As Integer
ReDim sColumnNames(dataRow.getColumnCount) As String
sColumnName = sColumnNames(nColumn)
For nColumn = 0 To dataRow.getColumnCount -1
sColumnName = sColumnNames(nColumn)
Next
```

GetColumnIndex

Gets the corresponding column index.

Syntax

```
Function getColumnIndex(columnName As String) As Integer
```

Parameters

- Column name

Results

Returns the corresponding column index.

Example

```
Dim nIndex As Integer
nIndex = dataRow.getColumnIndex("AddressLine1")
```

GetColumnCount

Gets the number of columns in the DataRow.

Syntax

```
Function getColumnCount() As Integer
```

Parameters

- None

Results

Returns the number of columns.

Example

```
Dim nColumnCount As Integer
nColumnCount = dataRow.getColumnCount()
```

GetByIndex

Gets the value from the field array by the column index in this DataRow.

Syntax

```
Function getByIndex(index As Integer) As String
```

Parameters

- Index with which the specified value is to be associated

Results

Returns the value for the column index in this DataRow. Returns empty string if the index is invalid.

Example

```
Dim sValue As String
sValue = dataRow.getByIndex(1)
```

GetByName

Gets the value from the field array by the column name in this DataRow.

Syntax

```
Function getByName(columnName As String) As String
```

Parameters

- Name with which the specified value is to be associated

Results

Returns the value for the column name in this DataRow; returns empty string if the column name does not exist.

Example

```
Dim sValue As String
sValue = dataRow.getByName("City")
```

Merge

Merges the given DataRow and the current DataRow.

Syntax

```
Sub merge(other As DataRow)
```

Parameters

- The other DataRow to be merged with the current DataRow

Results

None.

Example

```
Dim otherDataRow As New G1CLIENTlib.DataRow
...
dataRow.merge(otherDataRow)
```

SetByName

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

```
Sub setByName(columnName As String, value As String)
```

Parameters

- The name with which the specified value is to be associated
- Value to be associated with the specified name

Results

None.

Exceptions

- Blank column name
- Duplicate column name

Example

```
Dim newRow As G1CLIENTLib.DataRow
Set newRow= dataTable.netRow
newRow.setByName "AddressLine1", "100 Congress"
newRow.setByName "City", "Austin"
newRow.setByName "State", "Texas"
dataTable.addRow newRow
```

SetByIndex

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

```
Sub setByIndex(index As Integer, value As String)
```

Parameters

- The column index with which the specified value is to be associated
- Value to be associated with the specified name

Results

None.

Exceptions

- The column index is invalid

Example

```
Dim newRow As G1CLIENTLib.DataRow
Set newRow= dataTable.netRow
newRow.setByIndex 0, "100 Congress"
newRow.setByIndex 1, "Austin"
newRow.setByIndex 2, "Texas"
dataTable.addRow newRow
```

AddChild

Adds a new DataRow to the named parent/child relationship. If the named relationship exists, the supplied DataRow will be appended to the existing DataRow collection. Otherwise, a new collection will be created with the supplied DataRow as its only element.

Syntax

```
Sub addChild( childName As String, childDataRow As DataRow)
```

Parameters

- The name of the parent/child relationship (e.g., "Flood Plain Data," "References, " "Used By," etc.)
- The DataRow to be added to the relationship.

Results

None.

Example

```
Dim dataRow As New G1CLIENTLib.dataRow
Dim childDataRow As New G1CLIENTLib.dataRow

childDataRow .setByName "Address", "100 Congress"
childDataRow .setByName "City", "Austin"

dataRow.addChild "child1", dataRow
```

GetChildren

Retrieves the child rows from a named relationship.

Syntax

Function getChildren(childName As String) As DataRow()

Parameters

- The name of the parent/child relationship, e.g. "Flood Plain Data", "References", "Used By", etc.

Result

Returns the child rows from the named relationship.

Example

```
Dim dataRow As New G1CLIENTLib.dataRow
' Assume that dataRow has children ......
' Or more code to be needed
Dim rowsChild1() As Variant
rowsChild1 = dataRow.getChildren("child1")
```

ListChildNames

Retrieves all of the names of the named parent/child relationships.

Syntax

```
Function listChildNames() As String()
```

Parameters

None.

Results

Returns the set of the names of the named parent/child relationships.

Example

```
Dim dataRow As New G1CLIENTLib.dataRow
' Assume that dataRow has children ......
' Or more code to be needed
Dim sChildNames() As String
sChildNames = dataRow.listChildNames
```

SetChildren

Sets the rows of a supplied, named parent/child relationship. If rows previously existed under this name, they will be returned to the caller.

Syntax

```
Function setChildren(childName As String, DataRows As DataRow()) As
DataRow()
```

Parameters

None.

Results

Returns the set of the names of the named parent/child relationships.

Example

```
Dim dataRow1 As New G1CLIENTLib.dataRow
Dim dataRow2 As New G1CLIENTLib.dataRow
dataRow1.setByName "Address", "100 Congress"
dataRow1.setByName "City", "Austin"
dataRow2.setByName "Address", "200 Congress"
dataRow2.setByName "City", "Austin"

Dim rows(1) As G1CLIENTLib.dataRow

Set rows(0) = dataRow1
Set rows(1) = dataRow2

Dim newRows() As Variant
newRows = dataRowSpt.setChildren("child1", rows())
```

Map

Map is an object that maps keys to values. A map cannot contain duplicate keys — each key can map to at most one value.

Reset

Sets the cursor to be before the first map.

Syntax

Sub Reset()

Parameters

- None

Results

None.

Example

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

Next

Moves the cursor down one map from its current position.

Syntax

```
Sub Next()
```

Parameters

- None

Example

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
```

```
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

GetKey

Gets the key in the current map.

Syntax

```
Function getKey() As String
```

Parameters

- None

Results

Returns the key on the current map.

Example

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

GetValue

Gets the value in the current map.

Syntax

```
Function getValue() As String
```

Parameters

- None

Results

Returns the value on the current map.

Example

```
Dim requestMsg As New G1CLIENTLib.Message
Dim map As G1CLIENTLib.Map
Dim sKey As String
Dim sValue As String

requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_ID, "admin"
requestMsg.putContext requestMsg.CONTEXT_ACCOUNT_PASSWORD, "admin"

Set map = requestMsg.getContextMap

map.Reset
While (map.Next)
 sKey = map.getKey
 sValue = map.getValue
Wend
```

5

The Java API

In this section:

- Introduction 126
- Server 128
- Service 133
- Message 134
- DataTable 139
- DataRow 143

Introduction

A Java class is a blueprint or prototype that defines the variables and methods common to all objects of a certain type. A Java class also defines the implementation of a particular kind of object. It is with these classes that you can create Java applications. In general, Java objects are created from Java classes.

A Java object is a collection of related variables and methods written in the Java language utilizing the Java Virtual Machine (JVM). The data associated with a class or object is stored in variables. The behavior associated with a class or object is implemented with methods. Methods are similar to the functions or procedures in procedural languages such as C.

Java software objects interact and communicate with each other using messages. Additional information that the receiving object may need to perform its task is passed by parameters.

For more information on Java technology, go to **www.oracle.com/java**.

Constants

The Java API uses two sets of constants. The first set is for the Server component, described in the table below.

Table 21: Constants for the Server Component

Constant Name	Description	Example
Server.HOST	String for server host name. Default is "localhost".	65.89.200.89
Server.PORT	String for server port. Default is "8080".	10119
Server.ACCOUNT_ID	String for server account ID. Default is null.	user1
Server.ACCOUNT_PASSWORD	String for server account password. Default is null.	user1
Server.CONNECTION_TIMEOUT	String for server connection timeout, in millisecond. Default is "10000".	50000
Server.CONNECTION_TYPE	String for server connection type. Currently only support HTTP, HTTPS, or SOCKET. Default is "HTTP".	HTTP
Server.PROXY_HOST	String for proxy server host name. Default is null.	192.168.1.77
Server.PROXY_PORT	String for proxy server port. Default is null.	8080
Server.PROXY_USER	String for proxy server account ID. Default is null.	user1
Server.PROXY_PASSWORD	String for proxy server account password. Default is null.	user1

The second set of constants is for the Message component.

Table 22: Constants for the Message Component

Constant Name	Description/Default	Example
Message.CONTEXT_ACCOUNT_ID	String for message context account ID.	user1
Message.CONTEXT_ACCOUNT_PASSWORD	String for message context account password.	user1
Message.CONTEXT_SERVICE_NAME	String for message context service name.	echoservice

Error Messages

The Java API uses the following error messages:

- Error Messages for Connection
  - "Connection type not supported."
  - "Client timeout"
- Error Messages for creating DataTable:
  - "Blank column name"
  - "Duplicated column name"
  - "Index is out of bounds"
- Error Messages for Message Packaging Exception
  - "Cannot pack null Message"
  - "Input Message is null"
  - "Unable to connect to Server:"
  - "Failed to get Service"
  - "Unknown serialization type:"
  - "Unknown encoding type:"
  - "Gateway is not connected" (for SOCKET)

Example Application

The sample code shown below illustrates how to use the Java API.

```
try
{

    // Create Server
    Server server = new Server();

    // Set server connection properties
    server.setConnectionProperty(Server.HOST, "localhost");
    server.setConnectionProperty(Server.PORT, "10119");
    server.setConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
    server.setConnectionProperty(Server.ACCOUNT_ID, "guest");
    server.setConnectionProperty(Server.ACCOUNT_PASSWORD, "");

    // Connect to server
    server.connect();
```

```
    // Get Service From Server
    Service service = server.getService("ValidateAddress");

    // Create Input Message
    Message request = new Message();

    // Fill DataTable in the input message
    DataTable dataTable = request.getDataTable();
    DataRow row1 = dataTable.newRow();
    row1.set("AddressLine1", "4200 Parliament Place");
    row1.set("City", "Lanham");
    row1.set("StateProvince", "Maryland");
    dataTable.addRow(row1);
    DataRow row2 = dataTable.newRow();
    row2.set("AddressLine1", "100 Congress");
    row2.set("City", "Austin");
    row2.set("StateProvince", "Texas");
    dataTable.addRow(row2);

    // Set"option" Properties to the Input
    Message request.putOption("OutputCasing", "M");
    request.putOption("OutputRecordType", "A");

    // Process Input Message, return output Message
    Message reply = service.process(request);

    // Disconnect from server
    server.disconnect();

    // Get the result from the response message
    DataTable returnDataTable = reply.getDataTable();
    String[] columnNames = returnDataTable.getColumnNames();
    Iterator iter = returnDataTable.iterator();
    while (iter.hasNext())
    {
        DataRow row = (DataRow) iter.next();
       for (int col = 0; col & lt; returnDataTable.getColumnCount(); col++)

        {
            String value = row.get(columnNames[col]);
            System.out.println(value);
        }
    }
}
catch (Exception e)
{
    System.out.println("Error Occurred, " + e.getMessage());
}
```

Server

The Server class is used to connect to the server, disconnect from the server, and get the service from the server. The following table summarizes the functions each method performs in the Server class.

Connect

Reads the properties to determine which gateway connection to be used and makes a connection to the server. You can connect via HTTP, HTTPS, or SOCKET. However, HTTP and HTTPS do not actually connect to the server until a GetService or Process method is invoked. With a SOCKET connection type, the Connect method is fully functional.

Syntax

```
public void connect()
```

Parameters

None.

Results

Throws:

- ConfigurationException: When invalid configuration causes the inability to connect to the server. For example, an unknown protocol would cause a ConfigurationException. There is no value in attempting to retry connect() when this error occurs.
- ConnectionException: When unable to connect to the server. It might be possible to reconnect, depending on the underlying cause of the exception.
- MessageProcessingException: When an error occurs on the server that is not due to Configuration or Connection issues.

Example

```
Server server = new Server();

server.setConnectionProperty(Server.HOST, "localhost");
server.setConnectionProperty(Server.PORT, "10119");
server.setConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
server.setConnectionProperty(Server.ACCOUNT_ID, "guest");
server.setConnectionProperty(Server.ACCOUNT_PASSWORD, "");

try
{
 //Connect to server
 server.connect();
}
catch (ConfgurationException e)
{
 // indicate an error with configuration
}
catch (ConnectionException e)
{
 // handle connection issue (retry, report error, etc.)
}
catch (MessageProcessingException e)
{
 // report error
}
```

Connection Pooling

Connection pooling for the SOCKET connection type is available to the Java client. This section describes how to enable and disable connection pooling. By default connection pooling is disabled.

To enable connection pooling:

```
Server server = new Server();
Server.setConnectionProperty(Connection.SOCKET_POOL, "true");
```

To disable connection pooling:

```
Server server = new Server();
Server.setConnectionProperty(Connection.SOCKET_POOL, "false");
```

When connection pooling is enabled, the connect() method borrows a connection from the pool, and the disconnect() method returns the connection back to the pool. When pooling, the client must call disconnect() each time to return the connection to the pool.

Each thread should contain its own server, as shown in the following example:

```
{
 ...
 Server server = new Server();
 server.setConnectionProperty(Server.HOST, "localhost");
 server.setConnectionProperty(Server.PORT, "10119");
 server.setConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
 server.setConnectionProperty(Server.ACCOUNT_ID, "yourID");
 server.setConnectionProperty(Server.ACCOUNT_PASSWORD, "pwd");
 server.setConnectionProperty(Connection.SOCKET_POOL, "true");
 server.setConnectionProperty(Connection.SOCKET_POOL_MAX_ACTIVE, "20");
 server.setConnectionProperty(Connection.SOCKET_POOL_MIN_IDLE, "10");
 server.setConnectionProperty(Connection.SOCKET_POOL_MAX_TOTAL, "25");
 server.connect();
 ...
 service = server.getService(serviceName);
 reply = service.process(requestMessage);
 server.disconnect();
 ...
}
```

The following table lists the constants you can use for connection pooling.

Table 23: Constants for Connection Pooling

Constant Name	Description
SOCKET_POOL	Whether or not to use connection pooling if using the SOCKET connection type. Valid values are true or false. Default is false.
SOCKET_POOL_MAX_ACTIVE*	Maximum number of active socket connections that may be borrowed from the pool. Default is -1, which indicates no maximum.
SOCKET_POOL_MAX_IDLE*	Maximum number of idle socket connections remaining in the pool. Default is -1, which indicates no maximum.
SOCKET_POOL_MAX_TOTAL*	Maximum total number of pooled socket connections (both active and idle). Default is -1, which indicates no maximum.
SOCKET_POOL_MAX_WAIT*	Maximum amount of time (in milliseconds) to wait before throwing an exception when the pool is exhausted and the "when exhausted" action is WHEN_EXHAUSTED_BLOCK. Default is -1, which indicates no maximum.
SOCKET_POOL_MIN_EVICTABLE_IDLE_TIME_MILLIS*	Minimum amount of time a connection may sit idle in the pool before it is eligible

Constant Name	Description
	for eviction. Default is 1800000 (30 minutes).
SOCKET_POOL_MIN_IDLE*	Minimum number of connections allowed in the pool before the evictor thread (if active) creates new connections. Default is 0.
SOCKET_POOL_NUM_TESTS_PER_EVICTION_RUN*	Sets the number of idle connections to examine during each run of the evictor thread (if active). Default is -1, which indicates all idle connections are examined.
SOCKET_POOL_TEST_ON_BORROW*	Whether connections will be validated before being borrowed from the pool. Default is true.
SOCKET_POOL_TEST_ON_RETURN*	Whether connections will be validated before being returned to the pool. Default is false.
SOCKET_POOL_TEST_WHILE_IDLE*	Whether connections will be validated by the idle connection eviction thread. Default is false.
SOCKET_POOL_TIME_BETWEEN_EVICTION_RUNS_MILLIS*	Sets the number of milliseconds to sleep between runs of the idle connection evictor thread. When set to zero or a negative number, no idle connection evictor thread will be run. Default is 300000 (5 minutes).
SOCKET_POOL_WHEN_EXHAUSTED_ACTION*	Sets the "when exhausted action" to take when attempting to borrow a connection and none are available. Default is SOCKET_POOL_WHEN_EXHA USTED_BLOCK.
SOCKET_POOL_WHEN_EXHAUSTED_BLOCK*	A "when exhausted action" type indicating that when attempting to borrow a connection and none are available, the caller should block until a new object is available, or the maximum wait time has elapsed.
SOCKET_POOL_WHEN_EXHAUSTED_FAIL*	A "when exhausted action" type indicating that when attempting to borrow a connection and none are available, the caller should fail, throwing a ConnectionException.
SOCKET_POOL_WHEN_EXHAUSTED_GROW*	A "when exhausted action" type indicating that when attempting to borrow a connection and none are available, a new connection will be made anyway.

* Applicable only if using the SOCKET connection type and connection pooling is enabled.

Disconnect

Disconnects from the server.

Syntax

```
public void disconnect()
```

Parameters

None.

Results

Client is disconnected from the server.

Example

```
...
//Disconnect from server
server.disconnect();
```

SetConnectionProperty

Establishes the server connection configuration properties, such as host name and length of timeout.

Syntax

```
public void setConnectionProperty(String name, String value)
```

Parameters

- Name — the name of the connection property, such as HOST
- Value — the value for the name of the connection property, such as "www.myhost.com"

Results

None.

Exceptions

- ERROR-INVALID-COLUMN_NAME — an empty of null column name.
- ERROR_INVALID_VALUE — A null value.

Example

```
Server server = new Server();

server.setConnectionProperty(Server.HOST, "localhost");
server.setConnectionProperty(Server.PORT, "8080");

//Connect to server
server.connect();
```

GetService

Gets the service from the server.

Note: See the Component Reference section of this guide for a list of services that may be available to you.

Syntax

```
public Service getService(String serviceName)
```

Parameters

- Name - the name of the service

Results

Returns the specific service.

Exceptions

throws ServiceNotFoundException, ServiceCreationException

Example

```
Service service = server.getService("ValidateAddress");
```

Service

The Service class is used to process the message (i.e., send the message to the server and receive a response from the server).

Process

Process the input message and returns the response message.

Syntax

```
public Message process (Message message)
```

Parameters

- Input message

Results

Returns the response message.

Exceptions

- TimeoutException: When invalid configuration causes the inability to connect to the server. For example, an unknown protocol would cause a ConfigurationException. There is no value in attempting to retry connect() when this error occurs.
- ConnectionException: When unable to connect to the server. It might be possible to reconnect, depending on the underlying cause of the exception.
- MessageProcessingException: When an error occurs on the server that is not due to Configuration or Connection issues.

Example

```
try
{
 //Process Input Message, return output Message
 Message response = service.process(message);
}
catch (ConnectionException e)
{
 // handle connection issue (retry, report error, etc.)
}
catch (TimeoutException e)
{
 // handle timeout issue (retry, report error, etc.)
}
catch (MessageProcessingException e)
{
 // report error
}
```

Message

The Message class sends your input data and receives your output data from the service. The properties for Message include context properties, such as account ID, account password, service name, and service method; and option properties, which are the service-specific runtime options.

GetContext

Gets the value by the name in the "context" properties. Context properties include the following constants: account ID, account password, service name, service key, and request ID.

Syntax

```
public String getContext(String name)
```

Parameters

- Name - the name whose associated value is to be returned

Results

Returns the value associated with the name in the context properties. If the name does not exist, the method returns NULL.

Example

```
String value = message.getContext(Message.CONTEXT_ACCOUNT_ID);
```

GetContext

Gets the map that contains all of the context entries.

Syntax

```
public Map getContext()
```

Parameters

- None

Results

Returns the map that contains all of the context entries.

Example

```
Map context = message.getContext();
```

PutContext

Sets the value for the given name in the context properties. If there is an existing value present for the entity identified by the name, it is replaced. Context properties include the following constants: accountID, account password, service name, service key, and request ID.

Syntax

```
public void putContext(String name, String value)
```

Parameters

- Name - the name with which the specified value is to be associated
- Value - the value to be associated with the specified name

Results

None.

Example

```
message.putContext(Message.CONTEXT_ACCOUNT_ID, "user1");
```

PutContext

Adds the new context properties to the current context properties.

Syntax

```
public void putContext(Map map)
```

Parameters

- The new context hashtable to be added to the current context hashtable

Results

None.

Example

```
Map context = new HashMap();
...
message.putContext(context);
```

SetContext

Overwrites the current context properties with the new context properties.

Syntax

```
public void setContext(Map map)
```

Parameters

- The new context map that will replace the current context map.

Results

None.

Example

```
Map context = new Map ();
...
message.setContext(context);
```

GetOption

Gets the value by the name in the option properties. Option properties are the service-specific run-time options.

Syntax

```
public String getOption(String name)
```

Parameters

- Name - the name whose associated value is to be returned.

Results

Returns the value for the name in the option properties in the message OR NULL if the name does not exist.

Example

```
String value = message.getOption("OutputCasing");
```

GetOptions

Gets the map that contains all of the option entries.

Syntax

```
public Map getOptions();
```

Parameters

- None

Results

Returns the map that contains all of the option entries.

Example

```
Map options = message.getOptions();
```

PutOption

Sets the value for given name in the option properties. If there is an existing value present for the entity identified by the name, it is replaced. Option properties are the service specific run-time options.

Syntax

```
public void setOption(String name, String value)
```

Parameters

- Name - name with which the specified value is to be associated
- Value - value to be associated with the specified name

Results

None.

Example

```
message.setOption("OutputCasing", "M");
```

PutOptions

Adds the new option properties to the current option properties.

Syntax

```
public void putOptions(Map map)
```

Parameters

- The new option map to be added to the current option properties

Example

```
Map options = new HashMap();
...
message.putOptions(options);
```

SetOptions

Overwrites the current option properties with the new option properties.

Syntax

```
public void setOptions(Map map)
```

Parameters

- The new option map to replace the current option map

Results

None.

Example

```
Map options = new HashMap();
...
message.setOptions(options);
```

GetError

Gets the error message from the message.

Syntax

```
public String getError()
```

Parameters

- None

Results

Returns the error message in the message.

Example

```
String error = message.getError();
```

GetDataTable

Gets the data table in this message.

Syntax

```
public DataTable getDataTable()
```

Parameters

None.

Results

None.

Example

```
DataTable dataTable = message.getDataTable();
```

DataTable

DataTable contains the records for your input and output data. Using the methods associated with this class, you define the column names for your output and add records to the DataTable.

AddColumn

Adds the new column to the DataTable.

Syntax

```
public int addColumn(String columnName)
```

Parameters

- columnName

Results

Returns the index of the column

Example

```
DataTable dataTable = message.getDataTable();
int columnIndex = dataTable.addColumn("AddressLine1");
columnIndex = dataTable.addColumn("City")
```

GetColumnNames

Gets all the column names.

Syntax

```
public String[] getColumnNames()
```

Parameters

- None

Results

Returns the string array of column names.

Example

```
String[] columnNames = dataTable.getColumnNames();
```

GetColumnIndex

Gets the corresponding column index.

Syntax

```
public int getColumnIndex(String columnName)
```

Parameters

- Column name

Results

Returns the corresponding column index.

Example

```
int columnIndex = dataTable.getColumnIndex("City");
```

GetColumnCount

Gets the number of columns in the DataTable.

Syntax

```
public int getColumnCount()
```

Parameters

- None

Results

Returns the number of columns.

Example

```
int columnCount = dataTable.getColumnCount();
```

Clear

Clears the data in the DataTable.

Syntax

```
public void clear()
```

Parameters

- None

Results

None.

Example

```
dataTable.clear();
```

Iterator

An iterator that contains all DataRows in the DataTable.

Syntax

```
public Iterator iterator()
```

Parameters

- None

Results

Returns an iterator that contains all DataRows in the DataTable.

Example

```
Iterator iter = dataTable.iterator();
while (iter.hasNext())
{
 DataRow row = (DataRow)iter.next();
}
```

AddRow

Adds a row to the DataTable.

Syntax

```
public void addRow(DataRow row)
```

Parameters

- Row - DataRow to be added to the DataTable

Results

None.

Example

```
DataTable dataTable = message.getDataTable();

DataRow row = dataTable.newRow();
row.set("AddressLine1","4203 Greenridge");

dataTable.addRow(row);
```

NewRow

Creates a new DataRow to the DataTable.

Syntax

```
public DataRow newRow()
```

Parameters

- None

Results

Returns the newly created DataRow

Example

```
DataRow row = dataTable.newRow();
row.set("AddressLine1", "4203 Greenridge");

dataTable.addRow(row);
```

GetRowCount

Gets the number of DataRows in the DataTable.

Syntax

```
public int getRowCount()
```

Parameters

- None

Results

Returns the number of DataRows in the DataTable.

Example

```
int rowCount = dataTable.getRowCount();
```

Merge

Merges the given DataTable and the current DataTable.

Syntax

```
public void merge(DataTable other)
```

Parameters

- The other DataTable to be merged with the current DataTable

Results

None.

Example

```
DataTable otherDataTable = new DataTable();
dataTable.merge(otherDataTable);
```

DataRow

DataRow contains the individual records for your input and output data. Using the methods associated with this class, you define the column names for your output and add records to the DataTable.

GetColumnNames

Gets all the column names.

Syntax

```
public String[] getColumnNames()
```

Parameters

- None

Results

Returns the string array of column names.

Example

```
String[] columnNames = dataRow.getColumnNames();
```

GetColumnIndex

Gets the corresponding column index.

Syntax

```
public int getColumnIndex(String columnName)
```

Parameters

- Name - column name

Results

Returns the corresponding column index.

Example

```
int columnIndex = dataRow.getColumnIndex("City");
```

Get

Gets the value from the field array by the column index in this DataRow.

Syntax

```
public String get(int index)
```

Parameters

- Index with which the specified value is to be associated

Results

Returns the value for the column index in this DataRow.

Example

```
String value = dataRow.get(1);
```

Get

Gets the value from the field array by the column name in this DataRow.

Syntax

```
public String get(String columnName)
```

Parameters

- Name - name with which the specified value is to be associated

Results

Returns the value for the column name in this DataRow; returns empty string if the column name does not exist.

Example

```
String value = dataRow.get("City");
```

Merge

Merges the given DataRow and the current DataRow.

Syntax

```
public void merge(DataRow other)
```

Parameters

- The other DataRow to be merged with the current DataRow

Results

None.

Example

```
DataRow otherDataRow = new DataRow();
dataRow.merge(otherDataRow);
```

Set

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

```
public void set(int Index, String value)
```

Parameters

- The column index with which the specified value is to be associated
- Value to be associated with the specified name

Results

None.

Exceptions

- IndexOutOfBounds — the column index is invalid

Example

```
DataRow row = dataTable.newRow();
row.set(0,"4203 Greenridge");
row.set(1,"Austin");
row.set(2,"Texas")
dataTable.addRow(row);
```

AddChild

Adds a new DataRow to the named parent/child relationship. If the named relationship exists, the supplied DataRow will be appended to the existing DataRow collection. Otherwise, a new collection will be created with the supplied DataRow as its only element.

Syntax

```
public void addChild(String childName, DataRow childDataRow)
```

Parameters

- Name - the name of the parent/child relationship (e.g., "Flood Plain Data," "References, " "Used By," etc.)
- Value - the DataRow to be added to the relationship.

Results

None.

Example

```
DataRow childDataRow = new DataRow();
childDataRow.set("Address", "100 Congress");
...
DataRow dataRow = new DataRow();
...
dataRow.addChild("child1", childDataRow);
```

GetChildren

Retrieves the child rows from a named relationship.

Syntax

```
public List getChildren(String childName)
```

Parameters

- The name of the parent/child relationship, e.g. "Flood Plain Data", "References", "Used By", etc.

Results

Returns the child rows from the named relationship.

Example

```
List childRows = row.getChildren("child1");
```

ListChildNames

Retrieves all of the names of the named parent/child relationships.

Syntax

```
public Set listChildNames()
```

Parameters

None.

Results

Returns the set of the names of the named parent/child relationships.

Example

```
Set childNames = row.listChildNames();
```

SetChildren

Sets the rows of a supplied, named parent/child relationship. If rows previously existed under this name, they will be returned to the caller.

Syntax

```
public List setChildren(String childName, List DataRows)
```

Parameters

None.

Results

Returns the set of the names of the named parent/child relationships.

Example

```
List rows = dataRow.getChildren("child1");
parentRow.setChildren("child2", rows);
```

Set

Sets the value for the corresponding column for the DataRow. If the value for the name exists, the old value is replaced.

Syntax

```
public void set(int Index, String value)
```

Parameters

- The column index with which the specified value is to be associated

- Value to be associated with the specified name

Results

None.

Exceptions

- IndexOutOfBounds — the column index is invalid

Example

```
DataRow row = dataTable.newRow();
row.set(0,"4203 Greenridge");
row.set(1,"Austin");
row.set(2,"Texas")
dataTable.addRow(row);
```

6

ManagementAPI Methods

In this section:

- Introduction 150
- GetLicenseInfo 150
- GetVersionInfo 151

Introduction

There are two management API methods that are publicly available via the ManagementAPI Web service. Those methods include getLicenseInfo and getVersionInfo. The wsdl for the ManagementAPI Web service can be viewed using the following URL: `http://localhost:8080/managers/ManagementAPIService?wsdl`, where "localhost" is replaced by the name of the Spectrum™ Technology Platform server.

GetLicenseInfo

The method GetLicenseInfo returns a license object. The license object contains properties for machine type, operating system type, hostname, and CPU limit. It also contains an array of feature objects and an array of restriction objects. These arrays can be processed to determine specific information about features and restrictions. Feature has an ID, a name, and an enabled flag. Restriction has an ID, a limit, and a start date.

Web Service

ManagementAPIService

Parameters

None.

Result

Returns the license object.

Example

```
License
 string machineType
 string osType
 string hostName
 string CPULimit
 Feature[] features
 Restriction[] restrictions

Feature
 string ID;
 string name;
 Restriction[] restrictions

Restriction
 string ID
 long limit
 datetime startDate
 Feature[] features

ExpirationRestriction extends Restriction

UsageRestriction extends Restriction
 long usages
```

GetVersionInfo

The GetVersionInfo method returns an array of VersionInfo objects. A VersionInfo object has a name, version number, and a list of VersionAttribute objects. VersionAttribute objects are simple classes that have a label and a value. GetVersionInfo attributes are product-specific as the information is gathered and returned by the product itself. This same information is also displayed in the Version Information node of the Management Console.

Note: You must run GetVersionInfo once, see what values come back, and then parse the information to get specific pieces of information.

Web Service

ManagementAPIService

Parameters

None.

Result

Returns VersionInfo objects.

Example

```
VersionInfo
 string name
 string version
 VersionAttribute[] attributes

VersionAttribute
 string label
 string value
```

7

The .NET API

In this section:

- Introduction 154
- Server 156
- Service 159
- Message 159
- EnhancedDataTable 164

Introduction

.NET is a Microsoft® operating system platform that incorporates applications and a suite of tools and services which enhance Web service and application development.

The .NET framework uses components called Common Language Runtime (CLR), Framework Class Library (FCL), and ASP.NET. The CLR is equivalent to the Java Virtual Machine, in that it manages code and executes it in the native language of the machine on which it runs. The Framework Class Library is a massive library of re-usable object types that cover a myriad of program functions. ASP.NET is a server-side technology that allows web pages and services to load much faster than traditional ASP pages. Together, these three components of the .NET framework make application and Web development easier, more streamlined, and provides easier integration into existing environments. Clients and servers on different platforms running services written in various programming languages can communicate with each other swiftly and easily.

For more information on .NET technology, go to **msdn.microsoft.com/netframework**.

Constants

The .NET API uses two sets of constants. The first set is for the Server component, described in the table below.

Table 24: Constants for the Server Component

Constant Name	Description	Example
Server.HOST	String for server host name. Default is "localhost".	65.89.200.89
Server.PORT	String for server port. Default is "8080".	10119
Server.ACCOUNT_ID	String for server account ID. Default is null.	user1
Server.ACCOUNT_PASSWORD	String for server account password. Default is null.	user1
Server.CONNECTION_TIMEOUT	String for server connection timeout, in millisecond. Default is "10000".	50000
Server.CONNECTION_TYPE	String for server connection type. Currently only support HTTP, HTTPS, or SOCKET. Defaults is "HTTP".	HTTP(S)
Server.PROXY_HOST	String for proxy server host name. Default is null.	192.168.1.77
Server.PROXY_PORT	String for proxy server port. Default is null.	8080
Server.PROXY_USER	String for proxy server account ID. Default is null.	user1
Server.PROXY_PASSWORD	String for proxy server account password. Default is null.	user1

The second set of constants is for the Message component.

Table 25: Constants for the Message Component

Constant Name	Description	Example
Message.CONTEXT_ACCOUNT_ID	String for message context account ID.	user1
Message.CONTEXT_ACCOUNT_PASSWORD	String for message context account password.	user1
Message.CONTEXT_SERVICE_NAME	String for message context service name.	echoservice

Error Messages

The .NET API uses the following error messages:

- Error Messages for Connection
  - "Connection type not supported."
  - "Client timeout"
- Error Messages for Message Packaging Exception
  - "Input Message is null."

The other error messages will come from the .NET Framework Class Library if they do not get used properly.

Example Application

The sample code shown below illustrates how to use the .NET API.

```
using System;
using System.IO;
using System.Collections;
using System.Text;
using System.Data;
using glclient;

try
 {
 //Create Server
 Server server = new Server();

 //Set connect property to the server
 server.SetConnectionProperty(Server.HOST, "localhost");
 server.SetConnectionProperty(Server.PORT, "10119");
 server.SetConnectionProperty(Server.CONNECTION_TYPE, "SOCKET");
 server.SetConnectionProperty(Server.ACCOUNT_ID, "guest");
 server.SetConnectionProperty(Server.ACCOUNT_PASSWORD, "");

 //Connect to server
 server.Connect();

 //Get Service From Server
 Service service = server.GetService("ValidateAddress");

 //Create Input Message
 Message request = new Message();
```

```
//Fill dataTable in the input message
//Datatable is the .net Framework class
DataTable dataTable = request.GetDataTable();

DataColumn column1 = new DataColumn();
column1.DataType = System.Type.GetType("System.String");
column1.ColumnName = "AddressLine1";
dataTable.Columns.Add(column1);

DataColumn column2 = new DataColumn();
column2.DataType = System.Type.GetType("System.String");
column2.ColumnName = "City";
dataTable.Columns.Add(column2);

DataColumn column3 = new DataColumn();
column3.DataType = System.Type.GetType("System.String");
column3.ColumnName = "StateProvince";
dataTable.Columns.Add(column3);

DataRow newRow = dataTable.NewRow();
newRow[0]="4200 Parliament Place";
newRow[1]="Lanham";
newRow[2]="Maryland";

dataTable.Rows.Add(newRow);

//Set "option" Properties to the Input Message
request.PutOption("OutputCasing", "M");
request.PutOption("OutputRecordType", "A");

//Process Input Message, return output Message
Message reply = service.Process(request);

//Disconnect from server
server.Disconnect();

//Get the result from the response message
DataTable returnDataTable = reply.GetDataTable();

foreach(DataColumn dc in returnDataTable.Columns)
{
// more code to be added
string columnName = dc.ColumnName;
}
foreach(DataRow dr in returnDataTable.Rows)
{
for (int col = 0; col < returnDataTable.Columns.Count; col++)
{
// more code to be added
string value = (String)dr[col] ;
Console.WriteLine(value);
}
}
}
catch (Exception e)
{

//Error handling
Console.WriteLine("Error Ocurred, " + e.ToString());
}
```

Server

The Server class is used to connect to server, disconnect from the server, and get the service from the server.

Connect

Reads the properties to determine which gateway connection to be used and makes a connection to the server.

Note: .NET uses the HTTP, HTTPS, or SOCKET server connection protocol. HTTP and HTTPS logically establish a client connection, but do not actually connect to the server until a GetService or Process method is invoked. The SOCKET protocol establishes a connection to the server when Connect is invoked.

Syntax

```
public void Connect()
```

Parameters

None.

Results

None.

Exceptions

- "Connection type not supported."

Example

```
Server server = new Server();

// set connect property to the server
 server.SetConnectionProperty(Server.HOST, "localhost");
server.SetConnectionProperty(Server.PORT, "8080");
// more connection properties to be set
// Connect to server
server.Connect();
```

Disconnect

Disconnects from the server.

Syntax

```
public void Disconnect()
```

Parameters

None.

Results

Client is disconnected from the server.

Example

```
//Disconnect from server
server.Disconnect();
```

SetConnectionProperty

Establishes the server connection configuration properties, such as host name and length of timeout.

Syntax

```
public void SetConnectionProperty(String name, String value)
```

Parameters

- Name — the name of the connection property, such as HOST
- Value — the value for the name of the connection property, such as "www.myhost.com"

Results

None.

Example

```
Server server = new Server();

server.SetConnectionProperty(Server.HOST, "localhost");
server.SetConnectionProperty(Server.PORT, "8080");

//Connect to server
server.Connect();
```

GetService

Gets the service from the server.

Note: See the Component Reference section of this guide for a list of services that may be available to you.

Syntax

```
public Service getService(String serviceName)
```

Parameters

- Name - name of service

Results

Returns the specific service.

Example

```
Service service = server.GetService("ValidateAddress");
```

Service

The Service class is used to process the message (in other words, it sends the message to the server and receives a response from the server).

Process

Processes the input message and returns the response message.

Syntax

```
public Message Process(Message, message)
```

Parameters

- Input message

Results

Returns the response message.

Exceptions

MessageProcessingException

Example

```
//Process Input Message, return output Message
Message reply = service.Process(request);
```

Message

The Message class sends your input data and receives your output data from the service. The properties for Message include context properties, such as account ID, account password, service name, and service method; and option properties, which are the Service-specific runtime options.

GetContext

Gets the value by the name in the context properties. Context properties include the following constants: account ID, account password, service name, service key, and the request ID.

Syntax

```
public String GetContext(String name)
```

Parameters

None.

Results

Returns the value associated with the name in the "context" properties. If the name does not exist, the method returns NULL.

Example

```
String value = message.GetContext(Message.CONTEXT_ACCOUNT_ID);
```

GetContext

Gets the hashtable that contains all of the context entries. Hashtable is the .NET Framework class.

Syntax

```
public Hashtable GetContext()
```

Parameters

- None

Results

Returns the hashtable that contains all of the context entries.

Example

```
Hashtable context = message.GetContext();
```

PutContext

Sets the value for the given name in the context properties. If there is an existing value present for the entity identified by the name, it is replaced. Context properties include the following constants: account ID, account password, service name, service key, and request ID.

Syntax

```
public void PutContext(String name, String value)
```

Parameters

- Name - name with which the specified value is to be associated
- Value - value to be associated with the specified name

Example

```
message.PutContext(Message.CONTEXT_ACCOUNT_ID, "user1");
```

PutContext

Adds the new context properties to the current context properties.

Syntax

```
public void PutContext(Hashtable context)
```

Parameters

- The new context hashtable to be added to the current context hashtable

Results

None.

Example

```
//Hashtable is the .NET Framework class
Hashtable context = new Hashtable();
//more code
message.PutContext(context);
```

SetContexts

Overwrite the current context properties with the new context properties.

Syntax

```
public void SetContexts(Hashtable context)
```

Parameters

- Context - the new context hashtable that will replace the current context hashtable.

Results

None.

Example

```
//Hashtable is the .NET Framework class
Hashtable context = new Hashtable();
//more code
message.SetContexts(context);
```

GetOption

Gets the value by the name in the option properties. Option properties are the service-specific run-time options.

Syntax

```
public String GetOption(String name)
```

Parameters

- Name - the name whose associated value is to be returned.

Results

Returns the value for the name in the "option" properties in the message or NULL if the name does not exist.

Example

```
String value = message.GetOption("OutputCasing");
```

GetOptions

Gets the hashtable that contains all of the option entries. Hashtable is the .NET Framework class.

Syntax

```
public Hashtable GetOptions();
```

Parameters

- None

Results

Returns the hashtable that contains all of the option entries.

Example

```
Hashtable options = message.GetOptions();
```

PutOption

Sets the value for given name in the option properties. If there is an existing value present for the entity identified by the name, it is replaced. Option properties are the Service specific run-time options.

Syntax

```
public void PutOption(String name, String value)
```

Parameters

- Name - name with which the specified value is to be associated
- Value - value to be associated with the specified name

Example

```
message.PutOption("OutputCasing", "M");
```

PutOptions

Adds the new option properties to the current option properties.

Syntax

```
public void PutOptions(Hashtable options)
```

Parameters

- Option - the new option hashtable to be added to the current option hashtable

Results

None.

Example

```
//Hastable is the .NET Framework class
Hashtable options = new Hashtable();
// more code
message.PutOptions(options);
```

SetOptions

Overwrites the current option properties with the new option properties.

Syntax

```
public void SetOptions(Hashtable options)
```

Parameters

- Options - the new option hashtable to replace the current option hashtable

Results

None.

Example

```
//Hashtable is the .NET Framework class
Hashtable options = new Hashtable();
//more code
message.SetOptions(options);
```

GetError

Gets the error message from the message.

Syntax

```
public String GetError()
```

Parameters

- None

Results

Returns the error message in the message.

Example

```
String error = message.GetError();
```

GetDataTable

Gets the DataTable in this message. DataTable is .NET Framework class.

Syntax

```
public DataTable GetDataTable()
```

Parameters

None.

Results

None.

Example

```
//DataTable is the .net Framework class
DataTable dataTable = message.GetDataTable();

DataColumn column1 = new DataColumn();
column1.DataType = System.Type.GetType("System.String");
column1.ColumnName = "AddressLine1";
dataTable.Columns.Add(column1);

DataColumn column2 = new DataColumn();
column2.DataType = System.Type.GetType("System.String");
column2.ColumnName = "City";
dataTable.Columns.Add(column2);

DataRow newRow = dataTable.NewRow();
newRow[0]="4203 Greenridge";
newRow[1]="Austin";

dataTable.Rows.Add(newRow);
```

EnhancedDataTable

EnhancedDataTable is a class which extends .NET class DataTable.

AddChild

Adds a new DataRow to the named parent/child relationship. If the named relationship exists, the supplied DataRow will be appended to the existing DataRow collection. Otherwise, a new collection will be created with the supplied DataRow as its only element.

Syntax

```
public void AddChild(DataRow parentRow, string name, DataRow newChild)
```

Parameters

- Name - the name of the parent/child relationship (e.g., "Flood Plain Data," "References, " "Used By," etc.)
- DataRow - the DataRow to be added to the relationship.

Results

None.

Example

```
EnhancedDataTable dataTable = new EnhancedDataTable();

dataTable.Columns.Add(new DataColumn("AddressLine1",
System.Type.GetType("System.String")));
dataTable.Columns.Add(new DataColumn("City",
System.Type.GetType("System.String")));
dataTable.Columns.Add(new DataColumn("StateProvince",
System.Type.GetType("System.String")));
dataTable.Columns.Add(new DataColumn("PostalCode",
System.Type.GetType("System.String")));

DataRow row = dataTable.NewRow();

row[0] = "510 S Coit St";
row[1] = "Florence";
row[2] = "SC";
row[3] = "29501-5221";

EnhancedDataTable childDataTable = new EnhancedDataTable();

childDataTable.Columns.Add(new DataColumn("AddressLine2",
System.Type.GetType("System.String")));
childDataTable.Columns.Add(new DataColumn("City",
System.Type.GetType("System.String")));
childDataTable.Columns.Add(new DataColumn("StateProvince",
System.Type.GetType("System.String")));
childDataTable.Columns.Add(new DataColumn("PostalCode",
System.Type.GetType("System.String")));


DataRow childRow = childDataTable.NewRow();

childRow[0] = "241 Ne C St";
childRow[1] = "Willamina";
childRow[2] = "OR";
childRow[3] = "97396-2714";

dataTable.AddChild(row, "Child1", childRow);
dataTable.Rows.Add(row);
```

GetChildren

Retrieves the child rows from a named relationship.

Syntax

```
public EnhancedDataTable GetChildren(DataRow parentRow, string name)
```

Parameters

- ParentRow - the parent row
- Name - the name of the parent/child relationship, e.g. "Flood Plain Data", "References", "Used By", etc.

Results

Returns the child rows from the named relationship.

Example

```
EnhancedDataTable childRows = dataTable.GetChildren(parentRow, "child1");
```

ListChildNames

Retrieves all of the names of the named parent/child relationships.

Syntax

```
public string[] ListChildrenNames(DataRow parentRow)
```

Parameters

None.

Results

Returns the set of the names of the named parent/child relationships.

Example

```
string[] childNames = dataTable.ListChildrenNames( parentRow);
```

SetChildren

Sets the rows of a supplied, named parent/child relationship. If rows previously existed under this name, they will be returned to the caller.

Syntax

```
public void SetChildren(DataRow parentRow, string name, EnhancedDataTable
 newTable)
```

Results

Returns the set of the names of the named parent/child relationships.

Example

```
EnhancedDataTable childRows = dataTable1.GetChildren(parentRow, "child1");
dataTable2.SetChildren(otherParentRow, "child1", childRows);
```

8

Module Services

In this section:

- Address Now Module 168
- Enterprise Geocoding Module 204
- Enterprise Tax Module 354
- GeoConfidence Module 443
- Universal Addressing Module 446
- Universal Name Module 574

Address Now Module

Address Now Module

The Address Now Module is an address standardization and validation tool that provides comprehensive coverage for addresses outside the U.S. and Canada. Address Now is one of two address standardization and validation modules available for Spectrum™ Technology Platform. The other module is the Universal Addressing Module. The Address Now Module provides the following benefits over the Universal Addressing Module for addresses outside the U.S. and Canada:

- **Better data**—The database used by the Address Now Module is more up to date and complete in many countries than the database used by the Universal Addressing Module. This is because the Universal Addressing Module relies on data from Universal Postal Union (UPU), a body of the United Nations, for its international data and while the data coverage is extensive, the updates and the level of address details are not proactively managed by the UPU. Address Now, on the other hand, relies on data directly from the postal authorities (in most countries), plus other third-party data providers. This means that the data is more current with postal changes and is more detailed.
- **Drill-down feature**—The Address Now Module also offers drill-down capabilities to address data from any country, allowing users to rapidly enter address information without having to worry about the structure or making data entry mistakes.
- **Double-byte support**—The Address Now Module is Unicode enabled, recognizing Kanji and other double-byte characters.

Address Now Components

Address Now consists of the following components. These components can work with U.S., Canadian, and international addresses.

- **BuildGlobalAddresses**—Allows you to interactively build an address by searching for individual address elements.
- **GetGlobalCandidateAddresses**—Returns a list of addresses that are considered matches for a given address.
- **ValidateGlobalAddress**—Standardizes addresses using international postal data. ValidateGlobalAddress can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you have a significant amount of non-U.S. and non-Canadian address data, you should consider using ValidateGlobalAddress.

In cases where ValidateGlobalAddress returns multiple address matches for a given input address, you can use GetGlobalCandidateAddresses to return the address stack. GetGlobalCandidateAddresses is designed to return additional information from the postal databases to help you determine which of the returned addresses is the best match.

Address Now Database

The Address Now database contains postal data from all supported countries. You can install the entire database or the data for specific countries only. The database is installed on the server. This database is available by subscription from Pitney Bowes Software and is updated monthly.

BuildGlobalAddress

BuildGlobalAddress allows you to build a valid address starting with just a single address element or a few address elements. BuildGlobalAddress is part of the Address Now Module.

Using BuildGlobalAddress

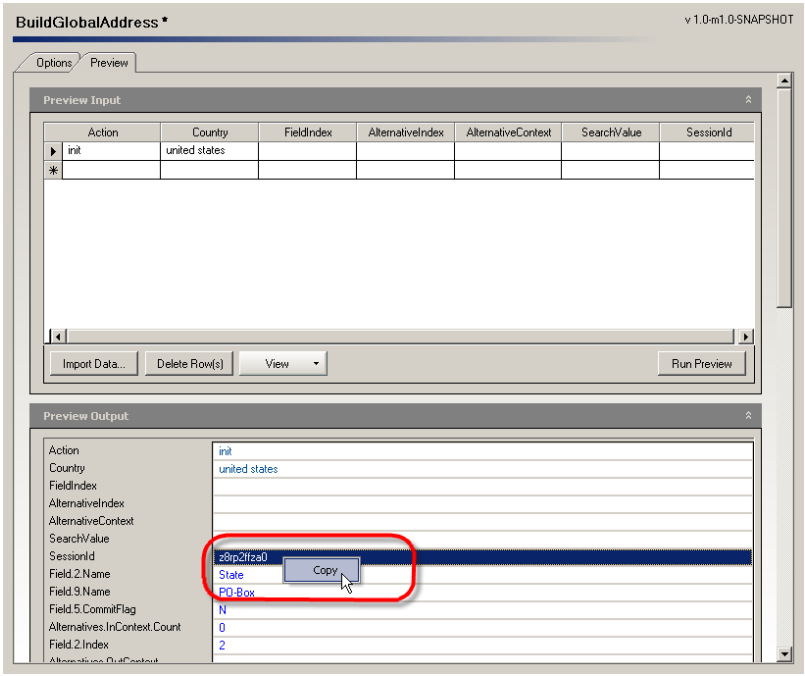
Building an address is an interactive process that requires you to select address elements at each step of the address building process. This means that building an address requires a sequence of calls to BuildGlobalAddress, not a single call. To start, you make an initialization call to BuildGlobalAddress. This call returns a session ID. You then use this session ID in subsequent calls. With each call, BuildGlobalAddress presents a list of alternative values for an address element. You select the value you want, then move on to the next address element until the complete address is built. With some exceptions, you need to make a separate call for each address element.

The overall process works like this:

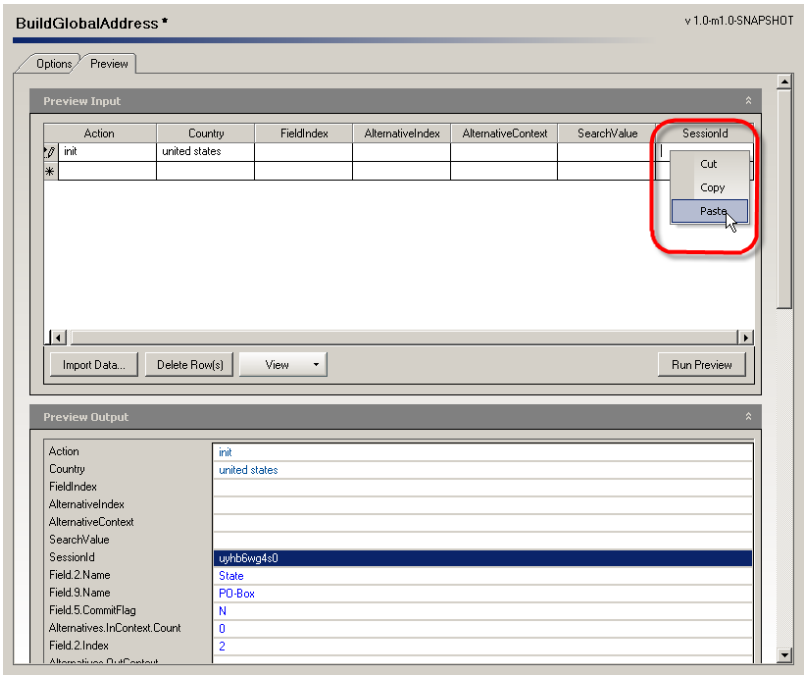
- First, you make an initialization call to open a session and receive a system-assigned session ID.
- Make a search call to find possible values for a given address element.
- When you have selected the value you want, you make a commit call to indicate the value you want for the given address element.
- Continue to make search/commit calls until all address elements are committed.
- Finally, you make a close call to end the session.

To familiarize yourself with how the process works, use the Management Console's Preview tab to step through the following procedure.

1. Open the Management Console.
2. Under the Services node, select **Build Global Address**.
3. On the **Options** tab, specify the options you want. For information on the options, see **Options** on page 173.
4. Click the **Preview** tab.
5. In the **Action** field type `init`.
6. In the **Country** field enter the country of the address you want to build.
7. Click **Run Preview**.
8. Under Preview Output, find the **SessionId** field, right-click the value, and select `Copy` from the pop-up menu.



9. Under Preview Input, right-click the **SessionId** field and select `Paste`.



10. Enter the following values in the input fields:

- Action—Type search.
- Country—Keep this field the same.
- FieldIndex—Type the index value of the first field you want to search. For example, if you know you want to search for an address in Chicago, you would type "1" because for U.S. addresses, field index 1 corresponds to the City field.
- SearchValue—Type the value you want to search for. For example, if you want to build an address in Chicago, you would type "chicago".
- SessionId—Keep the same value.

Note: The values in the other input fields are ignored.

11. Click **Run Preview** again.

12. The results of the search are placed in up to two output fields: **Alternatives.InContext** and **AlternativesOutContext**. For an explanation of the difference between in context results and out of context results, see **What Is Context?** on page 177.

13. When you have found the value you want, enter the following values in the input fields:

- **Action**—Type **commit**.
- **AlternativeIndex**—Type the index number for the alternative you choose. Index values start with 0, not 1. For example, if you search for Chicago the alternatives returned by BuildGlobalAddress would be indexed as follows. If you want to commit the value "CHICAGO" you would type "0" in the AlternativeIndex field.
  - 0—CHICAGO
  - 1—CHICAGO HTS
  - 2—CHICAGO PARK
  - 3—CHICAGO RIDGE
  - 4—EAST CHICAGO
  - 5—NORTH CHICAGO
  - 6—WEST CHICAGO
- **AlternativeContext**—Type `in` or `out` to indicate whether the index value you specified in **AlternativeIndex** is for the list of alternatives in the **Alternatives.InContext** field or the **Alternatives.OutContext** field.

- **SessionId**—Keep this value the same.

Note: The values in the other input fields are ignored.

14. Click **Run Preview** again. The value you specified will now be in the Field.n.Value field for the appropriate address element.
15. Repeat the search and commit steps as often as needed until you have built the address.
16. Close the session by entering the following values in the input fields:

- **Action**—Type **close**.
- **SessionId**—Keep this value the same.

Note: The values in the other input fields are ignored.

Input

Table 26: BuildGlobalAddress Input

columnName Parameter	Format	Description
Action	String	Specifies the action to take. One of the following: **init** Initialization. This action opens a session and returns a session ID which is required for all other actions. The init action requires the Country input field. **search** Searches for values for a specific address element and returns a list of alternative values for you to choose. The search action requires the following input fields: • FieldIndex • SearchValue • SessionId **commit** Assigns one of the values returned by the search action to the field. The commit action requires the following input fields: • AlternativeIndex • AlternativeContext • SessionId **clear** Un-commits the field specified in the FieldIndex field. The clear action requires the following input fields: • FieldIndex • SessionID **close** Ends a session. The close action requires the SessionId input field.
AlternativeContext	String	For the commit action, indicates whether you are choosing a value from the **Alternatives.InContext** field or the

columnName Parameter	Format	Description
		Alternatives.OutContext field. This field is ignored for other actions. One of the following: **in** You are committing a value from the **Alternatives.InContext** field. This means that the value you specify in the **AlternativeIndex** input field corresponds to a value in the **Alternatives.InContext** output field. **out** You are committing a value from the **Alternatives.OutContext** field. This means that the value you specify in the **AlternativeIndex** input field corresponds to a value in the **Alternatives.OutContext** output field.
AlternativeIndex	String [79]	For the commit action, specifies the value you want to use in the address you are building. For example, if you searched for a city and BuildGlobalAddress returns a list of three cities, you would indicate the city you want by specifying the index value for your choice. Index values for the alternatives presented by BuildGlobalAddress are zero-based, meaning that the first alternative has an index of 0, the second alternative has a value of 1, and so on. The input field is ignored for actions other than commit.
Country	String [79]	For the init action, specifies the country in which you want to build an address. Specify the country using the format you chose for input country format (English name two-character ISO 3116-1 Alpha-2 code, or three-character ISO 3116-1 Alpha-3 code). For a list of ISO codes, see **Country ISO Codes and Module Support**. This input field is ignored for actions other than init.
FieldIndex	String [79]	For the search action, specifies the address element that you want to search on. For the clear action, specifies the address element you want to un-commit. One of the following: **all** performs the "clear" action on all address elements. This option applies to the "clear" action only. **<IndexNumber>** Performs the action on a specific address element. To determine the index of an address element, first look at the Field.n.Name fields and locate the field you want. The value n indicates the field's index. For example, you want to look up ZIP Codes for U.S. addresses. After the init call you see that Field.0.Name is "Zip" indicating that the ZIP Code has a field index of "0". This input field is ignored for actions other than search and clear.

columnName Parameter	Format	Description
SearchValue	String [79]	For the search action, specifies the value you want to search for. This value must be appropriate for the field you specified in FieldIndex. For example, if you specified the ZIP Code field in FieldIndex, then you would enter a ZIP Code or partial ZIP Code in this field. Likewise if you chose the city field in FieldIndex you would specify a city name or partial city name in this field. if you leave the field blank the search will return all values that are in context. For more information about in context and out of context values, see **What Is Context?** on page 177. This input field is ignored for actions other than search.
SessionId	String [79]	Specifies the session ID you want to use for this call. To obtain a session ID use the init action. If a session is inactive for 5 minutes it will expire and you will need to perform a new init call to start a new session. This field is required for all actions except init.

Options

Table 27: BuildGlobalAddress Options

	Description
HomeCountry	Specifies the default country. You should specify the country where most of the addresses in your data are located. For example, if most of your addresses are in Canada, specify Canada. BuildGlobalAddress uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields.
OutputCountryFormat	Specifies the format to use for the country name in the output. One of the following: **E** The country in the output in English (default). **I** The country in the output as the two-character ISO code. **U** The country in the output as the three-character UPU code.
OutputPostalCodeSeparator	Specifies whether or not to use separators (spaces or hyphens) in ZIP Codes or Canadian postal codes. For example, a ZIP + 4® Code with the separator would be 20706-1844 and without the separator it would be 207061844. A Canadian postal code with the separator would be P5E"1S7 and without the separator it would be P5E1S7. **Y** Yes, use separator (default). **N** No, do not use separator. **Note:** Spaces are used in Canadian postal codes and hyphens in U.S. ZIP + 4® Codes.

	Description
ShowExtraAddressLine	Specifies whether or not to include the city, state/province, and postal code in one of the AddressLine output fields. Regardless of what you specify with this option, the output fields City, State/Province, and PostalCode will always contain the city, state/province, and postal code. **Y** Yes, include city, state/province, and postal code in an AddressLine output field (default). **N** No, do not include city, state/province, and postal code in an AddressLine output field.
MaximumResults	Allows you to set the default value for this option to any value from 1 to 10000; it has a default value of 50 records. Note that values set in Enterprise Designer override those set in Management Console

Output

BuildGlobalAddress returns address data and return codes for each input address.

Address Data

Table 28: BuildGlobalAddress Output

columnName	Format	Description
Action	String [79]	Shows the value specified in the Action input field for this call. For more information on this input field see **Input** on page 171.
AddressLine1	String [79]	The formatted first address line.
AddressLine2	String [79]	The formatted second address line.
AddressLine3	String [79]	The formatted third address line.
AddressLine4	String [79]	The formatted fourth address line.
AddressLine5	String [79]	The formatted fifth address line.
AddressLine6	String [79]	The formatted sixth address line.
AddressLine7	String [79]	The formatted seventh address line.
AddressLine8	String [79]	The formatted eighth address line.
AlternativeContext	String [79]	Shows the value specified in the AlternativeContext input field for this call. For more information, see **Input** on page 171.
AlternativeIndex	String [79]	Shows the value specified in the AlternativeIndex input field for this call. For more information on this input field see **Input** on page 171.
Alternatives.InContext	String [79]	A comma-delimited list of the possible values for the field you searched on which fit the context of fields you have already committed. For information on context see **What Is Context?** on page 177.

columnName	Format	Description
Alternatives.InContext.Count	String [79]	The number of "in context" results returned by your search. For information on context see **What Is Context?** on page 177.
Alternatives.OutContext	String [79]	A comma-delimited list of the possible values for the field you searched on which do not fit the context of fields you have already committed. For information on context see **What Is Context?** on page 177.
Alternatives.OutContext.Count	String [79]	The number of "out of context" results returned by your search. For information about context, see **What Is Context?** on page 177.
ApartmentLabel	String [79]	Apartment designator (such as STE or APT). For example: 123 E Main St. APT 3
ApartmentNumber	String [79]	Apartment number. For example: 123 E Main St. APT **3**
Building	String [79]	The name of a building.
City	String [79]	The city name.
Country	String [79]	Shows the value specified in the Country input field for this call. For more information about this input field, see **Input** on page 171.
Country	String [79]	The two- or three-character ISO code, or English name of the country. For a list of ISO codes, see **Country ISO Codes and Module Support**.
Department	String [79]	The name of a distinct part of anything arranged into divisions. For example, the Finance Department in a corporation.
Field.n.CommitFlag	String [79]	Indicates whether you have chosen a value for field n (i.e. "committed" a value). One of the following: **Y** Yes, the value of this field has been committed. **N** No, the value of this field has not been committed.
Field.n.Index	String [79]	An index value used to refer to field n, where n is 0 though 10. For example, for U.S. addresses the index value of the ZIP field is "0".
Field.n.Name	String [79]	The name of the address element contained in field n, where n is 0 through 10. For example, for U.S. addresses Field.0.Name is ZIP.
Field.n.Value	String [79]	The value that has been committed to field n, where n is 0 through 10. This field is blank on the init call.
FieldIndex	String [79]	Shows the value specified in the FieldIndex input field for this call. For more information on this input field see **Input** on page 171.
FirmName	String [79]	The name of a company. For example:

columnName	Format	Description
		Pitney Bowes Software 4200 PARLIAMENT PL STE 600 LANHAM MD 20706-1844 USA
HouseNumber	String [79]	House number. For example: **123** E Main St. Apt 3
POBox	String [79]	The post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode	String [79]	The postal code. In the U.S. this is the ZIP Code™.
PostalCode.AddOn	String [79]	The 4-digit add-on part of the ZIP + 4® Code. For example, in the ZIP Code™ 60655-1844, 1844 is the 4-digit add-on. (U.S. addresses only.)
PostalCode.Base	String [79]	The 5-digit ZIP Code™. For example 20706 (U.S. addresses only.)
Principality	String [79]	An area within a country. For example, England, Scotland, and Wales are principalities. This field will normally be blank.
SearchFieldIndex	String [79]	The index value of the field searched in the previous search action.
SearchValue	String [79]	Shows the value specified in the SearchValue input field for this call. For more information on this input field see **Input** on page 171.
SessionId	String [79]	Shows the value specified in the SessionId input field for this call. For more information on this input field see **Input** on page 171.
StateProvince	String [79]	The state or province abbreviation.
StreetName	String [79]	Street name. For example: 123 E **Main** St. Apt 3
StreetSuffix	String [79]	Street suffix. For example: 123 E Main **St**. Apt 3
SubCity	String [79]	A district or suburb. The subcity is used in countries where it is common to include the district or suburb within the address. For example, 27 Crystal Way **Bradley Stoke** Bristol BS32 8GA In this case, "Bradley Stoke" is the subcity.
SubStreet	String [79]	The second street address used to identify an address. Substreets are used in countries where it is common to give two street names in the address. For example, 12 The Mews **High Street**

columnName	Format	Description
		In this example, "High Street" is the substreet. Substreets can be used to precisely identify the delivery location. In the example, "The Mews" may be a small street that needs another street identification to properly locate the address, so "High Street" is included. In this case, "High Street" is the main or known street.
USCountyName	String [79]	For U.S. addresses, the name of the county where the address is located.

Return Codes

Table 29: BuildGlobalAddress Return Codes

columnName	Format	Description
Status	String [79]	Reports the success or failure of the match attempt. **null** Success **F** Failure
Status.Code	String [79]	Reason for failure, if there is one. • SessionError • SeverError • CountryNotFound
Status.Description	String [79]	Description of the problem, if there is one. **Please initialize new session** This value will appear if Status.Code=SessionError. **Null or empty action** This value will appear if Status.Code=SessionError. **Unknown action** This value will appear if Status.Code=SessionError. **Invalid session** This value will appear if Status.Code=SessionError. **Invalid value for** This value will appear if Status.Code=SessionError. **Cannot Search Committed Field** This value will appear if Status.Code=SessionError. **Module not licensed** This value will appear if Status.Code=ServerError. **Could Not Identify Country** This value will appear if Status.Code=CountryNotFound

What Is Context?

When you perform a search for an address element, BuildGlobalAddress looks at the address elements that you have already committed and splits up the values it returns based on whether or not the returned

values exist within the context of the address elements you have already committed. For example, in the U.S. the following cities exist:

In Illinois:

- CHICAGO
- CHICAGO HTS
- CHICAGO RIDGE
- NORTH CHICAGO
- WEST CHICAGO

In Indiana:

- EAST CHICAGO

In Nevada:

- CHICAGO PARK

If you have already committed a value of "IN" (Indiana) for the state and then searched for the city "chicago", BuildGlobalAddress would return EAST CHICAGO as an "in context" result because it exists in Indiana, and it would return all the other matches for "chicago" as out-of-context results. Likewise, if you committed a value of "IL" (Illinois) for the state, BuildGlobalAddress would return EAST CHICAGO and CHICAGO PARK as out of context, and CHICAGO, CHICAGO HTS, CHICAGO RIDGE, NORTH CHICAGO, and WEST CHICAGO as "in context."

GetGlobalCandidateAddresses

GetGlobalCandidateAddresses returns a list of addresses that are considered matches for a given input address. If the input address matches multiple addresses in the Address Now database, the possible matches are returned. If the input address matches only one address in the Address Now database, no address data is returned.

GetGlobalCandidateAddresses is part of the Address Now Module.

Input

GetGlobalCandidateAddresses takes a standard address as input. All addresses use this format no matter what country the address is from. AddressLine1 and Country are required input fields. The other fields are optional.

Table 30: GetGlobalCandidateAddresses Input

columnName	Format	Description
AddressLine1	String [79]	First address line. This is a required field.
AddressLine2	String [79]	Second address line
AddressLine3	String [79]	Third address line
AddressLine4	String [79]	Fourth address line
AddressLine5	String [79]	Fifth address line

columnName	Format	Description
AddressLine6	String [79]	Sixth address line
AddressLine7	String [79]	Seventh address line
AddressLine8	String [79]	Eighth address line
City	String [79]	City name
StateProvince	String [79]	State or province.
PostalCode	String [10]	The postal code for the address in one of these formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999
Country	String	The country. Specify the country using the format you chose for input country format (English name or ISO code). For a list of ISO codes, see **Country ISO Codes and Module Support**.
FirmName	String [79]	Company or firm name

Options

Table 31: GetGlobalCandidateAddresses Options

optionName	Description/Valid Values
HomeCountry	Specifies the default country. Specify the country that is the destination of most of your mailpieces. For example, if most of your mailpieces are going to Canada, specify Canada. GetGlobalCandidateAddresses uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields. For a list of ISO codes, see **Country ISO Codes and Module Support**.
OutputCountryFormat	Specifies the format to use for the country name in the output. One of the following: **E** The country in the output in English (default). **I** The country in the output in the two-character ISO code. **U** The country in the output in the three-character UPU code.
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output in mixed case (default). For example: 123 Main St Mytown FL 12345

optionName	Description/Valid Values
	U The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345
OutputPostalCodeSeparator	Specifies whether or not to use separators (spaces or hyphens) in ZIP™ Codes or Canadian postal codes. For example, a ZIP + 4® Code with the separator would be 20706-1844 and without the separator it would be 207061844. A Canadian postal code with the separator would be P5E"1S7 and without the separator it would be P5E1S7. **Y** Yes, use separator (default) **N** No, do not use separator **Note:** Spaces are used in Canadian postal codes and hyphens in U.S. ZIP + 4® Codes.
ShowExtraAddressLine	Specifies whether or not to include the city, state/province, and postal code in one of the AddressLine output fields. Regardless of what you specify with this option, the output fields City, State/Province, and PostalCode will always contain the city, state/province, and postal code. **Y** Yes, include city, state/province, and postal code in an AddressLine output field (default). **N** No, do not include city, state/province, and postal code in an AddressLine output field.
MaximumResults	The maximum number of candidate addresses to output. The default is 50. The maximum value is 100.
ReturnUserData	Specifies whether or not to include in the output data from the input address that could not be validated. **Y** Yes, include input data that could not be validated. **N** No, do not include input data that could not be validated (default).

Output

GetGlobalCandidateAddresses returns address data and return codes for each address.

Address Data

Table 32: GetGlobalCandidateAddresses Address Data Output

columnName	Format	Description
AddressLine1	String [79]	The formatted first address line.
AddressLine2	String [79]	The formatted second address line.
AddressLine3	String [79]	The formatted third address line.
AddressLine4	String [79]	The formatted fourth address line.

columnName	Format	Description
AddressLine5	String [79]	The formatted fifth address line.
AddressLine6	String [79]	The formatted sixth address line.
AddressLine7	String [79]	The formatted seventh address line.
AddressLine8	String [79]	The formatted eighth address line.
ApartmentLabel	String [79]	Apartment designator (such as STE or APT). For example: 123 E Main St. **APT** 3
ApartmentNumber	String [79]	Apartment number. For example: 123 E Main St. APT **3**
Building	String [79]	The name of a building.
City	String [79]	The city name.
Country	String [79]	The ISO code or English name of the country. For a list of ISO codes, see **Country ISO Codes and Module Support**.
Department	String [79]	The name of a distinct part of anything arranged into divisions. For example, the Finance Department in a corporation.
FirmName	String [79]	The name of a company. For example: **Pitney Bowes Software** 4200 PARLIAMENT PL STE 600 LANHAM MD 20706-1844 USA
HouseNumber	String [79]	House number. For example: **123** E Main St. Apt 3
POBox	String [79]	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode	String [79]	The postal code as required by the local postal authority. For example, in the U.S. the postal code is the ZIP Code.
PostalCode.AddOn	String [79]	For U.S. addresses, the last four digits of the ZIP + 4® Code.
PostalCode.Base	String [79]	For U.S. addresses, the five-digit ZIP Code.
Principality	String [79]	An area within a country. For example, England, Scotland, and Wales are principalities. This field will normally be blank.
StateProvince	String [79]	The state or province abbreviation.
StreetName	String [79]	Street name. For example: 123 E **Main** St. Apt 3

columnName	Format	Description
StreetSuffix	String [79]	Street suffix. For example: 123 E Main **St**. Apt 3
SubCity	String [79]	A district or suburb. The subcity is used in countries where it is common to include the district or suburb within the address. For example, 27 Crystal Way **Bradley Stoke** Bristol BS32 8GA In this case, "Bradley Stoke" is the subcity.
SubStreet	String [79]	The second street address used to identify an address. Substreets are used in countries where it is common to give two street names in the address. For example, 12 The Mews **High Street** In this example, "High Street" is the substreet. Substreets can be used to precisely identify the delivery location. In the example, "The Mews" may be a small street that needs another street identification to properly locate the address, so "High Street" is included. In this case, "High Street" is the main or known street.
USCountyName	String [79]	For U.S. addresses, the name of the county where the address is located.

Return Codes

Table 33: GetGlobalCandidateAddresses Return Codes

columnName	Format	Description
ACRCode	String [79]	The Address Correction Result (ACR) code describes what data has been changed in each record. For information on what this code means, see **The ACR Code** on page 202.
Confidence	String [79]	The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct.
Status	String [79]	Reports the success or failure of the match attempt. **Null** Success **F** Failure
Status.Code	String [79]	Reason for failure, if there is one. • RequestFailed • ServerError • CountryNotFound
Status.Description	String [79]	Description of the problem, if there is one. **Maximum records cannot be set to 0. Minimum value should be 1** This value will appear if Status.Code=RequestFailed.

columnName	Format	Description	
		Address Not Found	This value will appear if Status.Code=RequestFailed.
		Module not licensed	This value will appear if Status.Code=ServerError.
		Could Not Identify Country	This value will appear if Status.Code=CountryNotFound.

ValidateGlobalAddress

ValidateGlobalAddress provides enhanced address standardization and validation for addresses outside the U.S. and Canada. ValidateGlobalAddress can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you need to validate addresses outside the U.S. and Canada, you should consider using ValidateGlobalAddress.

ValidateGlobalAddress is part of the Address Now Module.

Input

ValidateGlobalAddress takes a standard address as input. All addresses use this format no matter what country the address is from.

Table 34: ValidateGlobalAddress Input

columnName	Format	Description
AddressLine1	String [79]	First address line
AddressLine2	String [79]	Second address line
AddressLine3	String [79]	Third address line
AddressLine4	String [79]	Fourth address line
AddressLine5	String [79]	Fifth address line
AddressLine6	String [79]	Sixth address line
AddressLine7	String [79]	Seventh address line
AddressLine8	String [79]	Eighth address line
City	String [79]	City name
StateProvince	String [79]	State or province.

columnName	Format	Description
PostalCode	String [79]: 99999 ~~99999999~~ A9A9A9 A9A 9A9 9999 999	The postal code for the address. In the U.S. this is the ZIP Code™.
Country	String [79]	Specify the country using the format you chose for input country format (English name or ISO code). For a list of ISO codes, see **Country ISO Codes and Module Support**.
FirmName	String [79]	Company or firm name

Options

Input Data Options

Table 35: ValidateGlobalAddress Input Data Options

optionName	Description
HomeCountry	Specifies the default country. You should specify the country where most of the addresses are located. For example, if most of the addresses you process are in Canada, specify Canada. ValidateGlobalAddress uses the home country to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields. For a list of valid values, see **Country ISO Codes and Module Support**.

Output Data Options

Table 36: ValidateGlobalAddress Output Data Options

optionName	Description
OutputCountryFormat	Specifies the format to use for the country name in the output. One of the following: **E** The country in the output is English (default). **I** The country in the output in the two-character ISO code. **U** The country in the output in the three-character UPU code.
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345

optionName	Description
OutputPostalCodeSeparator	Specifies whether to use a separator (spaces or hyphens) in ZIP™ Codes or Canadian postal codes. For example, a ZIP + 4® Code with the separator would be 20706-1844 and without the separator it would be 207061844. A Canadian postal code with the separator would be P5E"1S7 and without the separator it would be P5E1S7. **Y** Yes, use separator (default) **N** No, do not use separator **Note:** Spaces are used in Canadian postal codes and hyphens in U.S. ZIP + 4® Codes.
ShowExtraAddressLine	Specifies whether to include the city, state/province, and postal code in one of the AddressLine output fields. Regardless of what you specify with this option, the output fields City, State/Province, and PostalCode will always contain the city, state/province, and postal code. **Y** Yes, include city, state/province, and postal code in an AddressLine output field (default). **N** No, do not include city, state/province, and postal code in an AddressLine output field.
StandardizeAddressOnFail	Specifies whether to return a standardized address when an address cannot be validated. The address is formatted using the preferred address format for the address's country. If this option is not selected, the output address component fields (StreetName, HouseNumber, etc.) are blank when address validation fails. **N** No, do not format failed addresses (default). **Y** Yes, standardize failed addresses.
FormatOnFail	Specifies whether to return a formatted address when an address cannot be validated. The address is formatted using the preferred address format for the address's country. **Y** Yes, return a formatted address when an address cannot be validated. **N** No, do not return a formatted address when an address cannot be validated (default).
ValidateAddress	Enables address validation. Address validation does the following: • Matches components to the relevant country's reference data • Corrects spelling errors • Adds missing components • Corrects or adds postal codes **Y** Yes, validate addresses (default). **N** No, do not validate addresses.
FormatAddress	Formats the address components into the statutory postal or custom formats. **Y** Yes, format addresses (default). **N** No, do not format addresses.

Standardization Options

Table 37: ValidateGlobalAddress Standardization Options

optionName	Description	
StandardizeComponent.Department	Specifies whether or not to populate the Department field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.FirmName	Specifies whether or not to populate the FirmName field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.Building	Specifies whether or not to populate the Building field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.SubBuilding	Specifies whether or not to populate the SubBuilding field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.HouseNumber	Specifies whether or not to populate the HouseNumber field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.SubStreet	Specifies whether or not to populate the SubStreet field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.StreetName	Specifies whether or not to populate the StreetName field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.POBox	Specifies whether or not to populate the POBox field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.SubCity	Specifies whether or not to populate the SubCity field when standardizing an address.	
	Y	Yes (default)
	N	No

optionName	Description	
StandardizeComponent.City	Specifies whether or not to populate the City field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.USCountyName	Specifies whether or not to populate the USCountyName field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.StateProvince	Specifies whether or not to populate the StateProvince field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.Principality	Specifies whether or not to populate the Principality field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.PostalCode	Specifies whether or not to populate the PostalCode field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.Plus4	Specifies whether or not to populate the +4 field when standardizing an address.	
	Y	Yes (default)
	N	No
StandardizeComponent.Country	Specifies whether or not to populate the Country field when standardizing an address.	
	Y	Yes (default)
	N	No
ReportVulgarWords	Specifies whether or not to look for vulgar words. If this option is enabled, ValidateGlobalAddress returns a value in the WCRCode output field to indicate the results.	
	Y	Yes
	N	No (default)
FlagVulgarWords	Specifies whether or not to mark vulgar words in the output using the format ">VulgarWord<".	
	Y	Yes
	N	No (default)
DebugOutput	This option controls whether or not to include troubleshooting information in the output fields Email1, Email2, URL1, and URL2.	
	Y	Yes

optionName	Description	
	N	No (default)

Validation Options

Table 38: ValidateGlobalAddress Validation Options

optionName	Description	
ValidateComponent.Department	Specifies whether or not to include the Department field when validating an address.	
	Y	Yes
	N	No (default)
ValidateComponent.FirmName	Specifies whether or not to include the FirmName field when validating an address.	
	Y	Yes
	N	No (default)
ValidateComponent.Building Option.ValidateComponent.Building	Specifies whether or not to include the Building field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.SubBuilding	Specifies whether or not to include the SubBuilding field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.HouseNumber	Specifies whether or not to include the HouseNumber field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.SubStreet	Specifies whether or not to include the SubStreet field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.StreetName	Specifies whether or not to include the StreetName field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.POBox	Specifies whether or not to include the POBox field when validating an address.	

optionName	Description	
	Y	Yes (default)
	N	No
ValidateComponent.SubCity	Specifies whether or not to include the SubCity field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.City	Specifies whether or not to include the City field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.USCountyName	Specifies whether or not to include the USCountyName field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.StateProvince	Specifies whether or not to include the StateProvince field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.Principality	Specifies whether or not to include the Principality field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.PostalCode	Specifies whether or not to include the PostalCode field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.Plus4	Specifies whether or not to include the +4 field when validating an address.	
	Y	Yes (default)
	N	No
ValidateComponent.Country	Specifies whether or not to include the Country field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.Department	Specifies whether or not to correct the Country field when validating an address.	
	Y	Yes (default)

optionName	Description	
	N	No
ForceUpdate.FirmName	Specifies whether or not to correct the FirmName field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.Building	Specifies whether or not to correct the Building field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.SubBuilding	Specifies whether or not to correct the SubBuilding field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.HouseNumber	Specifies whether or not to correct the HouseNumber field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.SubStreet	Specifies whether or not to correct the SubStreet field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.StreetName	Specifies whether or not to correct the StreetName field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.POBox	Specifies whether or not to correct the POBox field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.SubCity	Specifies whether or not to correct the SubCity field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.City	Specifies whether or not to correct the City field when validating an address.	
	Y	Yes (default)
	N	No

optionName	Description	
ForceUpdate.USCountyName	Specifies whether or not to correct the USCountyName field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.StateProvince	Specifies whether or not to correct the StateProvince field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.Principality	Specifies whether or not to correct the Principality field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.PostalCode	Specifies whether or not to correct the PostalCode field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.Plus4	Specifies whether or not to correct the +4 field when validating an address.	
	Y	Yes (default)
	N	No
ForceUpdate.Country	Specifies whether or not to correct the Country field when validating an address.	
	Y	Yes (default)
	N	No
ReplaceAlias.Department	Specifies whether or not to overwrite the Department field if an alias is found in the Address Now database.	
	Y	Yes
	N	No (default)
ReplaceAlias.FirmName	Specifies whether or not to overwrite the FirmName field if an alias is found in the Address Now database.	
	Y	Yes
	N	No (default)
ReplaceAlias.Building	Specifies whether or not to overwrite the Building field if an alias is found in the Address Now database.	
	Y	Yes
	N	No (default)

optionName	Description
ReplaceAlias.SubBuilding	Specifies whether or not to overwrite the SubBuilding field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.HouseNumber	Specifies whether or not to overwrite the HouseNumber field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.SubStreet	Specifies whether or not to overwrite the SubStreet field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.StreetName	Specifies whether or not to overwrite the StreetName field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.POBox	Specifies whether or not to overwrite the POBox field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.SubCity	Specifies whether or not to overwrite the Subcity field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.City	Specifies whether or not to overwrite the City field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.USCountyName	Specifies whether or not to overwrite the USCountyName field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.StateProvince	Specifies whether or not to overwrite the StateProvince field if an alias is found in the Address Now database. **Y** Yes **N** No (default)

optionName	Description
ReplaceAlias.Principality	Specifies whether or not to overwrite the Principality field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.PostalCode	Specifies whether or not to overwrite the PostalCode field if an alias is found in the Address Now database. **Y** Yes (default) **N** No
ReplaceAlias.Plus4	Specifies whether or not to overwrite the +4 field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
ReplaceAlias.Country	Specifies whether or not to overwrite the Country field if an alias is found in the Address Now database. **Y** Yes **N** No (default)
CautiousUpdate	This option, used in conjunction with the "Force Update", ensures that no major changes are made to the data during processing. **Y** Yes **N** No (default)
CrossComponentMatch	Specifies whether or not to correct common address standardization and validation errors by performing cross-component matching. Cross-component matching checks for matches between data found in one field in the input data and another field in the Address Now database. **Y** Yes **N** No (default)
UseReferenceDiacritics	Specifies whether or not ValidateGlobalAddress modifies the address to match the diacritics (accents, umlauts, etc.) in the postal database when the only changes to the address are the diacritics . One of the following: **Y** Yes (default) **N** No For example, if **UseReferenceDiacritics** is enabled, the following would occur: Input City: Chalon-Sur-Saône City in the postal database: CHALON SUR SAONE Output City: CHALON SUR SAONE Input City: ARTEMIVS'K City in the postal database: ARTEMIVSK Output City: ARTEMIVSK

optionName	Description
	If **UseReferenceDiacritics** is not enabled, the following would occur: Input City: Chalon-Sur-Saône Reference City: CHALON SUR SAONE Output City: Chalon-Sur-Saône Input City: ARTEMIVS'K Reference City: ARTEMIVSK Output City: ARTEMIVS'K Note that this option has no effect on the **Transliteration** option.
KeepStandardizationChanges	Specifies whether or not standardization changes such as changing "ROAD" to "RD" should be reported in the ACR code. **Y** Yes **N** No (default)
AcceptanceLevel	The Acceptance Level setting specifies the minimum number of address components that must be validated in order for the whole address to be considered validated. The value specified for AcceptanceLevel corresponds to the second character of the ACR code. For more information, see **The ACR Code** on page 202. The acceptance level differs from the InnerMatchScore option in that acceptance level measures how many components Validate Global Address validated, regardless of how well the validated components matched to address components in the postal databases, whereas InnerMatchScore indicates the probability that the output address is the correct, validated version of the input address. One of the following: **-1** The acceptance level is automatically set to an appropriate level based on the address's country. For example, U.S. addresses are processed with an acceptance level of 4. **0** No components validated (default) **1** Country only validated **2** City and country validated **3** City, postal code and country validated **4** Street, city, postal code and country validated **5** Premise number, building name, sub-building, PO box, company, street, city, postal code, and country validated
InnerMatchScore	Specifies the minimum confidence level for address validation. Addresses with a value in the Confidence output field greater than or equal to this value is validated, and those that have a lower value will not be validated (the output field Status will contain F.) Specify any value between 0 and 100. The higher the value, the higher the degree of confidence necessary for effective address validation. The default is 60.
CompanyWeight	A whole number from 0 to 10, indicating the relative importance of the FirmName field compared to the data in the Address Now database.

optionName	Description
	This affects the confidence value, and can be used to tailor the confidence to distinguish correct and incorrect updates. For more information, see **The ACR Code** on page 202. The default value is 1.
StreetWeight	A whole number from 0 to 10, indicating the relative importance of the StreetName field compared to the data in the Address Now database. A whole number from 0 to 10, indicating the relative importance of this field compared to the others. For more information, see **The ACR Code** on page 202. The default value is 10.
CityWeight	A whole number from 0 to 10, indicating the relative importance of the City field compared to the data in the Address Now database. A whole number from 0 to 10, indicating the relative importance of this field compared to the others. For more information, see **The ACR Code** on page 202. The default value is 8.
PostcodeWeight	A whole number from 0 to 10, indicating the relative importance of the PostalCode field compared to the data in the Address Now database. A whole number from 0 to 10, indicating the relative importance of this field compared to the others. For more information, see **The ACR Code** on page 202. The default value is 8
OuterMatchScoreLines	A value from 0 to 8 indicating the number of address lines to use when calculating the outer match score. The default is 8. For more information on the outer match score, see **The Outer Match Score** on page 201.

Output Format Options

Table 39: ValidateGlobalAddress Output Format Options

optionName	Description
FormatComponent.Department	Specifies whether or not the Department field should be included in the output of a formatted address. **Y** Yes (default) **N** No
FormatComponent.FirmName	Specifies whether or not the FirmName field should be included in the output of a formatted address. **Y** Yes (default) **N** No
FormatComponent.Building	Specifies whether or not the Building field should be included in the output of a formatted address. **Y** Yes (default) **N** No

optionName	Description	
FormatComponent.SubBuilding	Specifies whether or not the SubBuilding field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.HouseNumber	Specifies whether or not the HouseNumber field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.SubStreet	Specifies whether or not the SubStreet field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.StreetName	Specifies whether or not the StreetName field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.POBox	Specifies whether or not the POBox field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.SubCity	Specifies whether or not the SubCity field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.City	Specifies whether or not the City field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.USCountyName	Specifies whether or not the USCountyName field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.StateProvince	Specifies whether or not the StateProvince field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No
FormatComponent.Principality	Specifies whether or not the Principality field should be included in the output of a formatted address.	
	Y	Yes (default)
	N	No

optionName	Description
FormatComponent.PostalCode	Specifies whether or not the PostalCode field should be included in the output of a formatted address. **Y** Yes (default) **N** No
FormatComponent.Plus4	Specifies whether or not the +4 field should be included in the output of a formatted address. **Y** Yes (default) **N** No
FormatComponent.Country	Specifies whether or not the Country field should be included in the output of a formatted address. **Y** Yes **N** No (default)
Transliteration	Specifies how to format diacritics in the output address. One of the following: **0** No transliteration is performed. Diacritic characters are left as specified in the input and/or postal database. Default. **1** Diacritic characters are removed and replaced with the equivalent unadorned character. **2** Diacritic characters are transliterated to an equivalent unadorned character or character sequence using language-specific transliteration rules. For example, the following shows the effect of each of the three transliteration options on a Swedish address. Note the differences in "Västra Frölunda". **0** Gustaf Wernersgata 12 S-42132 Västra Frölunda **1** Gustaf Wernersgata 12 S-42132 Vastra Frolunda **2** Gustaf Wernersgata 12 S-42132 Vaestra Froelunda

Output

Address Data Output

Table 40: ValidateGlobalAddress Address Data Output

columnName	Format	Description
AddressLine1	String [79]	The formatted first address line.
AddressLine2	String [79]	The formatted second address line.

columnName	Format	Description
AddressLine3	String [79]	The formatted third address line.
AddressLine4	String [79]	The formatted fourth address line.
AddressLine5	String [79]	The formatted fifth address line.
AddressLine6	String [79]	The formatted sixth address line.
AddressLine7	String [79]	The formatted seventh address line.
AddressLine8	String [79]	The formatted eighth address line.
ApartmentLabel	String [79]	Apartment designator (such as STE or APT). For example: 123 E Main St. APT 3
ApartmentNumber	String [79]	Apartment number. For example: 123 E Main St. APT **3**
Building	String [79]	The name of a building.
City	String [79]	The city name.
Country	String [79]	The ISO code or English name of the country. For a list of ISO codes, see **Country ISO Codes and Module Support**.
Department	String [79]	A subdivision of a country used in French and Spanish speaking countries. For example, France is divided into 100 departments.
FirmName	String [79]	The name of a company. For example: **Pitney Bowes Software** 4200 PARLIAMENT PL STE 600 LANHAM MD 20706-1844 USA
HouseNumber	String [79]	House number. For example: **123** E Main St. Apt 3
Latitude	String [79]	The most precise latitude that could be determined for the address. This could be a point level location or a centroid. The level of precision can be determined by looking at the ECRCode output field. For more information, see **The ECR Code** on page 201.
Longitude	String [79]	The most precise longitude that could be determined for the address. This could be a point level location or a centroid. The level of precision can be determined by looking at the ECRCode output field. For more information, see **The ECR Code** on page 201.
POBox	String [79]	The post office box number. If the address is a rural route address, the rural route box number will appear here.

columnName	Format	Description
PostalCode	String [79]	The postal code. In the U.S. this is the ZIP Code™.
PostalCode.AddOn	String [79]	The 4-digit add-on part of the ZIP + 4® Code. For example, in the ZIP Code™ 60655-1844, 1844 is the 4-digit add-on. (U.S. addresses only.)
PostalCode.Base	String [79]	The 5-digit ZIP Code™. For example 20706 (U.S. addresses only.)
Principality	String [79]	An area within a country. For example, England, Scotland, and Wales are principalities. This field will normally be blank.
StateProvince	String [79]	The state or province abbreviation.
StreetName	String [79]	Street name. For example: 123 E **Main** St. Apt 3
StreetSuffix	String [79]	Street suffix. For example: 123 E Main **St**. Apt 3
SubCity	String [79]	A district or suburb. The subcity is used in countries where it is common to include the district or suburb within the address. For example, 27 Crystal Way **Bradley Stoke** Bristol BS32 8GA In this case, "Bradley Stoke" is the subcity.
SubStreet	String [79]	The second street address used to identify an address. Substreets are used in countries where it is common to give two street names in the address. For example, 12 The Mews **High Street** In this example, "High Street" is the substreet. Substreets can be used to precisely identify the delivery location. In the example, "The Mews" may be a small street that needs another street identification to properly locate the address, so "High Street" is included. In this case, "High Street" is the main or known street.
USCountyName	String [79]	For U.S. addresses, the name of the county where the address is located.

Return Codes

Table 41: ValidateGlobalAddress Return Codes

columnName	Format	Description
ACRCode	String [79]	The Address Correction Result (ACR) code describes what data has been changed in each record. For information on what this code means, see **The ACR Code** on page 202.

columnName	Format	Description
Confidence	String [79]	The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct. This value is the same as the last three digits of the ACR code, referred to as the validation match score. For more information, see **The ACR Code** on page 202.
ECRCode	String [79]	The Enhanced Correction Result (ECR) code describes the level of precision of the latitude and longitude returned for the address. For mroe information, see **The ECR Code** on page 201.
Email1	String [79]	Extra standardization information.
Email2	String [79]	Extra standardization information.
OuterMatchScore	String [79]	A score that measures changes to each address line. For more information, see **The Outer Match Score** on page 201.
Status	String [79]	Reports the success or failure of the match attempt. • **null**—Success • **F**—Failure
Status.Code	String [79]	Reason for failure, if there is one. • UnableToValidate • ServerError • CountryNotFound
Status.Description	String [79]	Description of the problem, if there is one. • **Address Not Found**—This value will appear if Status.Code=UnableToValidate. • **Module not licensed**—This value will appear if Status.Code=ServerError. • **Could Not Identify Country**—This value will appear if Status.Code=CountryNotFound.
URL1	String [79]	Extra standardization information.
URL2	String [79]	Extra standardization information.
WCRCode	String [79]	The Word Correction Result (WCR) code describes vulgar words found in the input address. The code has two components: • **Location code**—One of the following: • **AB**—Indicates a vulgarity was found in the address. • **NB**—Indicates a vulgarity was found in the name. • **Count**—The number of vulgar words found in the location indicated by the location code. For example, AB2 indicates that two vulgar words were found in the input address.

The ECR Code

The Enhanced Correction Result (ECR) code describes the level of precision of the latitude/longitude coordinates returned for the address. The code consists of a prefix followed by a dash then the body of the code.

The prefix always begins with "EL" followed by a number from 1 to 5 indicating the overall level of precision:

- 5—Point geocode
- 4—Street centroid
- 3—Postcode centroid
- 2—City centroid
- 1—Region centroid

The body of the code identifies the components that were used to match the address to a geocode. Note that the body consists of letters and numerals. Here's what they mean:

- P—Premise/house number, building or PO box
- S—Street
- T—City
- R—Region/state
- Z—Postal code
- C—Country

There are only two numeric options in the body: 4 or 0

- 4—The component data was available to make the geocode-address match.
- 0—The component date was not available.

For example: EL4-P0S4T4R4Z4C4

In this example, the 0 following the P tells us that premise/house number data was not available to make this address match; everything from street to country, however, was used in the assignment.

The Outer Match Score

The outer match score indicates how much ValidateGlobalAddress changed each address line to validate the address. The score compares the address lines before standardization and after validation and formatting. This score is only generated if you set the option `OuterMatchScoreLines` to a value greater than 0.

The outer match score is similar to the validation match score, which is part of the ACR code (see **The ACR Code** on page 202). The difference is that the outer match score measures any change to an address line, including formatting, whereas the validation match score measures only whether or not the data could be validated,

For example, take the following input address lines before processing:

Address Line 1: 5 camden cres
Address Line 2: bath
Address Line 3: uk

After processing the address lines are:

Address Line 1: 5 Camden Crescent
Address Line 2: Bath
Address Line 3: BA1 5HY
Address Line 4: United Kingdom

This has a validation match score of 84% and in outer match score of 23%.

The validation match score is high because the address components were fairly accurate before validation. The street name was valid except for casing and use of an abbreviation. The city and country were both

valid. The only thing not correct was the postal code (in this case it was missing). Hence the relatively high validation match score of 84%.

The outer match score is low because after formatting, the address lines are considerably different from the input. In this case, Address Line 3 contained "uk" on input, and contains "BA1 5HY" on output. Line 4 was empty on input, and in populated on output. Address line 1 has also changed. The outer score is therefore quite low.

The ACR Code

The Address Correction Result (ACR) code describes what data has been changed in each record. An example of an ACR is:

L5-P0S0A5T1R0Z0C4-098

ACR codes consist of three parts:

- Validation Level
- Component Status
- Validation Match Score

Validation Level

The first two characters of the address correction result state the type and level of validation.

The first character, which is always alphabetic, specifies the type of validation:

- **U**—Unable to standardize address
- **C**—Address is in component form
- **L**—Address has been formatted into address lines
- **R**—Address has been reverted and has not reached acceptable level

The second character, which is always numeric, specifies the level of validation. The higher the level, the better the validation will be. The levels that can be achieved are as follows:

- **0**—No components validated
- **1**—Country only validated
- **2**—City and country validated
- **3**—City, postal code and country validated
- **4**—Street, city, postal code and country validated
- **5**—Premise number, building name, sub-building, PO box, company, street, city, postal code, and country validated.

Component Status

The second part of the ACR code gives the status of the main address components. The address components are identified as follows:

- Character 3-4: P—Premise/house number
- Character 5-6: S—Street
- Character 7-8: A—Subcity (city area)
- Character 9-10: T—City
- Character 11-12: R—Region/state
- Character 13-14: Z—Postal code/ZIP Code®
- Character 15-16: C—Country

A number follows each component and can take one of the following values:

- **0**—Not found/empty
- **1**—Derived using position in input data

- **2**—Recognized using the Address Now Module database
- **3**—Recognized and updated to standard form using the Address Now Module database
- **4**—Validated using Address Now Module database
- **5**—Updated/corrected using Address Now Module database
- **6**—Added using Address Now Module database
- **7**—Correctly empty
- **8**—Partial recognition using Address Now Module database
- **9**—Needs correcting to match Address Now Module database

Validation Match Score

The Validation Match Score comprises characters 17-19, the final three digits of the ACR code. This is a comparison between the standardized data (in component format) and the suggested match returned from the Address Now Module database.

This score is calculated by examining all fields returned from the Address Now Module database and comparing them individually with the existing component data. The overall match score is then calculated by combining these individual values into an average score, taking into account the match score weightings, which can be set from the address validation options dialog box. For example,

Input data:

AddressLine1: 11 High Street
City: Anytown
Country: UK

Standardized data:

Premise: 11
Street: High Street
City: Anytown

When validated, the data returned from the Address Now Module database for this record may be:

Premise: 11
Street: High Street
City: Anytown
Postal Code: ZZ9 9ZZ

Comparing the Address Now Module database to the standardized data we get:

- Premise: 100% match
- Street: 100% match
- City: 100% match
- Postal Code: not used, because empty on input

Combining these percentages gives us a match score of 100%.

Another example may be:

Input data:

AddressLine1: bergerstrasse 12
AddressLine2: munich
AddressLine3: 80124
Country: Germany

Standardized data:

Premise: 12
Street: Bergerstr.
City: München
Postal Code: 80124

Address Now Module database output:

Premise: 12
Street: Burgerstr.
City: München
Postal Code: 80142

Comparing the Address Now Module database output to the standardized data we get:

- Premise:100% match
- Street: 90% match (the actual figure is determined by a textual comparison of the two values)
- City: 100% match
- Postal Code: 80% match (because the numbers are transposed)

This gives an overall match score of 92% if the match score weightings are all set at 1. Increasing the match score weighting of the postal code will decrease the match score, because the postal code component score (80%) will be made more important in the calculation. Increasing the match score weighting of the city will increase the match score, because the city component score (100%) will be made more important.

For example:

L5-P4S4A5T5R4Z4C4-098

- L shows that formatting has been carried out to create the address lines
- The validation level is 5, meaning that the highest level of matching against the Address Now Module database was attained
- All component codes except subcity (A) and city (T) are set to 4 indicating that they were validated using the Address Now Module database
- The subcity code and city code are set to 5 indicating that these components were corrected using the Address Now Module database

The overall address matched the Address Now Module database at 98%.

Note: You may also receive a value of "SDS" for the Validation Match Score. A return of SDS indicates that the address has not been standardized, possibly as a result of the address being reverted.

Enterprise Geocoding Module

Enterprise Geocoding Module

The Enterprise Geocoding Module performs address standardization, address geocoding, and postal code centroid geocoding. You can enter an address and get outputs such as geographic coordinates, which can be used for detailed spatial analysis and demographics assignment. You can also enter a geocode, a point represented by a latitude and longitude coordinate, and receive address information about the provided geocode.

Enterprise Geocoding Stages

Enterprise Geocoding Module consists of the following components. The specific components you have depend on your license.

- **GeocodeAddressAUS**—Takes an address in Australia and returns latitude/longitude coordinates and other information.
- **GeocodeAddressGBR**—Takes an address in Great Britain and returns latitude/longitude coordinates and other information.

- **GeocodeAddressGlobal**—Takes an address in any supported country and returns latitude/longitude coordinates and other information. This component will only geocode addresses from countries you have licensed. It does not support Australia and Great Britain.
- **Geocode Address**—Takes an address located in any of the supported countries and returns the city centroid or, for some countries, postal centroid. The World component cannot geocode to the street address level.
- **Geocode Africa** — Provides street-level geocoding for many African countries. It can also determine city or locality centroids, as well as postal code centroids for selected countries.
- **Geocode Middle East** — Provides street-level geocoding for many Middle East countries. It can also determine city or locality centroids. The Middle East component supports both English and Arabic character sets.
- **Geocode Latin America** — Provides street-level geocoding for many Latin American countries. It can also determine city or locality centroids. There is postal code coverage for selected countries.
- **GeocodeUSAddress**—Takes an input address and returns latitude/longitude coordinates and other address information.
- **GNAFPIDLocationSearch**—Identifies the address and latitude/longitude coordinates for a Geocoded National Address File Persistent Identifier (G-NAF PID).
- **ReverseAPNLookup**—Takes an Assessor's Parcel Number (APN), Federal Information Processing Standards (FIPS) county code, and FIPS state code and returns the address of the parcel.
- **ReverseGeocodeUSLocation**—Takes as input a geocode (latitude and longitude coordinate) and returns the address of the location.

Enterprise Geocoding Databases

The following Enterprise Geocoding Module databases are installed on the Spectrum™ Technology Platform server. Some of the databases are available by subscription from Pitney Bowes Software and are updated monthly or quarterly. Others are licensed from the USPS®.

U.S. Geocoding Databases (U.S. Only)

These databases contain the spatial data necessary to perform address standardization and geocoding. You must install at least one of these databases to perform geocoding for USA. You set the database that you want to match against with the processing options. Enterprise Geocoding tries to match to the database you indicate. To verify you are matching to the database you want, you can review the value returned in the StreetDataType output field.

These databases use proprietary files called GSD files. For ZIP Code centroid matching, the file us.Z9 contains all the centroid info for all states and normally has a z9 extension.

- **Centrus Enhanced Geocoding**—This database consists of TIGER data provided by the U.S. Geological Survey and address data provided by the U.S. Postal Service.
- **TomTom Geocoding**—This database provides more up-to-date data than the Centrus Enhanced Geocoding database. It requires an additional license. This data is provided by TomTom, a third-party provider of spatial data, and postal data from the U.S. Postal Service.
- **NAVTEQ Geocoding**—This database provides more up-to-date data than the Centrus Enhanced Geocoding database. It requires an additional license. NAVTEQ data is provided by NAVTEQ, a third-party provider of spatial data. For more information about these databases, contact your sales representative.
- **ZIP + 4 Centroid**—This database provides only address standardization and ZIP + 4 centroid matching. It does not provide street-level matching.

Each geocoding database has an optional Statewide Intersections Index. The Statewide Intersection Index is designed to enable fast intersection identification on a statewide basis. For example, the Statewide Intersection Index will allow the database search for "1st and Main St, CO" and return a list of possible matches in Colorado more quickly than searching the entire geocoding database for each instance of the intersection.

U.S. Points Databases (U.S. Only)

Points databases contain data for locating the center of a parcel. These databases provides enhanced geocoding accuracy for internet mapping, property and casualty insurance, telecommunications, utilities, and others.

These databases are optional, but either Centrus Enhanced Points or Centrus Premium Points is required for Reverse Assessor's Parcel Number (APN) Lookup. These databases are also separately licensed.

- **Centrus Points**—This database contains the data necessary to locate the center of a parcel or building. It does not contain Assessor's Parcel Number (APN) or elevation data.
- **Centrus Elevation**—This database contains the same data as Centrus Points, plus elevation data.
- **Centrus Enhanced Points**—This database contains the same data as Centrus Points, plus APN data.
- **Centrus Premium Points**—This database contains the same data as Centrus Points, plus both APN and elevation data.
- **Centrus TomTom Points Database**—The data in this database is provided by TomTom, a third-party provider of spatial data.
- **Master Location Data** — This database provides the best available address point location for every mailable and deliverable address in the United States.

Reverse Geocoding Database (U.S. Only)

This database contains the data you need to convert a latitude/longitude location to an address.

This database is optional, but is required for ReverseGeocodeUSThis database is also separately licensed.

Auxiliary Files (U.S. Only)

Auxiliary files contain user-defined records. You can use auxiliary files to provide custom data to use in address matching and geocode matching.

DPV® Database (U.S. Only)

The Delivery Point Validation database allows you to check the validity of any individual mailing address in the U.S. The DPV database is distributed as an optional feature and can be installed to enhance the geocoding database's ability to validate mailing addresses. Each time an edition of the geocoding database is released, a corresponding edition of the optional DPV database is released. The date of the DPV database must match the date of the geocoding database for DPV processing to function. DPV lookups may not be performed after the expiration date of the DPV database.

This database is optional, but is required for CASS Certified™ processing. The DPV database is also required to determine ZIP + 4 and ZIP + 4 related output (DPBC, USPS record type, etc.). This database is also separately licensed.

Note: Postal Service licensing prohibits using DPV for the generation of addresses or address lists, and also prohibits the DPV database being exported outside the United States.

EWS Database (U.S. Only)

The Early Warning System (EWS) database contains data that prevents address records from miscoding due to a delay in postal data reaching the U.S. Postal database.

The USPS® refreshes the EWS file on a weekly basis. Unlike the DPV and LACS^{Link} databases, the EWS database does not need to have the same date as the geocoding database. You can download the EWS.zip file free of charge from the CASS section of the USPS® RIBBS website at:

https://ribbs.usps.gov//index.cfm?page=doclist

When you download the EWS database, you will receive a file named OUT. You must rename the OUT file to EWS.txt before using it.

LACS Link Database (U.S. Only)

LACS Link Database (U.S. Only)

The LACS^{Link} database allows you to correct addresses that have changed as a result of a rural route address converting to street-style address, a PO Box renumbering, or a street-style address changing.

This database is optional, but is required for CASS Certified™ processing. The LACS^{Link} database is also required in CASS mode to receive ZIP + 4 and ZIP + 4 related output (delivery point barcode, USPS record type, etc.).

The date of the LACS^{Link} database must match the date of the geocoding database for LACS^{Link} processing to function.

Note: USPS licensing prohibits using LACS^{Link} for the generation of addresses or address lists, and also prohibits the LACS^{Link} database being exported outside the United States.

International Geocoding Databases

International geocoding databases contain the spatial data necessary to perform address standardization and geocoding for locations outside the U.S. Each country has its own database, and some countries have optional databases that provide enhanced geocoding.

United Kingdom AddressBase Plus Database

AddressBase Plus is a point database sourced from Ordnance Survey®, Royal Mail, and local authorities.

The AddressBase Plus database provides the highest level of precision, as reflected in S8 result codes. The database includes objects without postal addresses, such as subdivided properties, places of worship and community centers.

The AddressBase Plus database is built around the UPRN (Unique Property Reference Number). The UPRN is the unique identifier that provides a persistent reference to a unique property, regardless of changes in the property name, status, subdivision, use (such as from single occupancy to multiple occupancy) or demolition of the property. All historic, alternative, and provisional addresses are recorded against the same UPRN. The UPRN is returned with every AddressBase Plus candidate, except for Northern Ireland addresses.

Since the Ordnance Survey data source does not contain addresses for Northern Ireland, AddressBase Plus is supplemented with Royal Mail® postcode address data for Northern Ireland. This Northern Ireland data has postcode centroid (result code S3) precision only.

For more information on the AddressBase Plus, see the **AddressBase Plus Support Page** from Ordnance Survey.

United Kingdom Address-Point Database

The Address-Point database contains street addresses and is source from Ordnance Survey®. Candidate location will be usually to a resolution of 0.1 meters from the building/parcel delivery point. This database is suitable for applications that demand this highest available level of precision, including routing applications.

The Address-Point database provides a greater level of geocoding precision than can be delivered by the CodePoint database. This high level of precision is reflected in S8 and S7 result codes.

While the Ordnance Survey data source does not contain addresses for Northern Ireland, the Address-Point dataset from Pitney Bowes Software is supplemented with Royal Mail® postcode address data for Northern Ireland. This Northern Ireland data has postcode centroid (result code S3) precision only.

For more information on the Ordnance Survey Address-Point data source, see:

http://www.ordnancesurvey.co.uk/oswebsite/products/address-point/index.html

United Kingdom CodePoint Database

The CodePoint Postal Address File (PAF) database provides postcode centroid geocoding. The CodePoint database is suitable for most applications involving address matching, validation, etc.

The CodePoint database is sourced from Royal Mail and covers street addresses for the UK (Great Britain and Northern Ireland). The CodePoint database is licensed for the entire dataset, rather than by region. The postcode centroid precision provided by the CodePoint database is reflected in S3 result codes.

For more information on the Royal Mail data source, see:

www.royalmail.com

Australia Geocoded National Address File (G-NAF)

This database provides enhanced geocoding for Australian addresses. This is the only authoritative Australian national index of locality, street and number, validated with geographic coordinates. It contains both officially recognized rural and urban addresses and unofficial addresses (aliases). Postal addresses and PO Boxes are not included. However, because some rural areas do not have adequate rural address information, roadside mail box (RMB) numbers, Lot numbers, and Block & Section numbers have been included in the G-NAF data set.

When you install this database there will be two subfolders:

- **GNAF123**—Contains the point-level dictionary. This has the highest precision of geocoding (characterized by Reliability Level 1, 2, or 3.)
- **GNAF456** —Contains the remainder of address information in G-NAF that does not meet high precision geocoding criteria (characterized by Reliability Level 4, 5, or 6.)

You must specify each of these as separate database resources in the Management Console.

We recommend that you use both databases to validate the existence of addresses but only use the GANF123 for parcel-level geocoding. If you do not require parcel-level geocodes you can use the GANF456 database for geocoding.

New Zealand Point Database

The New Zealand Point Database is based on postal point data which has a roof top precision point of each unique street address. Location X and Y returned for candidates from this database are roof top precision.

This data is maintained by the government authority, Land Information New Zealand. This database is a monthly update from what the local district councils supply.

Other Point Databases for International Countries

A number of other point databases are available for international countries. As of 2014 Q2, the Enterprise Geocoding Module point databases are available for the following countries:

- Canada
- Mexico
- India
- Japan
- Singapore
- France
- Germany
- Sweden

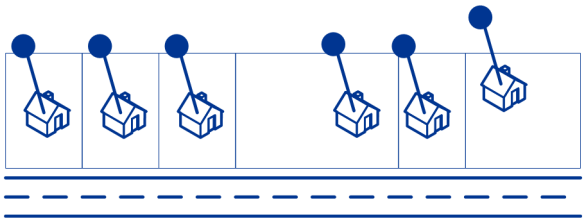
Additional point databases will be available in future releases. Contact your sales representative for information on licensing the Enterprise Geocoding Module point databases.

Geocoding Concepts

Geocoding is the process of determining the latitude/longitude coordinates of an address. There are different ways that an address can be geocoded. In order of most accurate to least accurate, these methods are:

Point Level Matching

Point-level matching locates the center of the actual building footprint or parcel. This is the most accurate type of geocode and is used in industries such as internet mapping, insurance, telecommunications, and utilities.

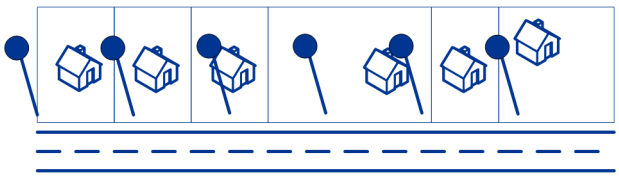


Centerline matching is used with point-level matching to tie a point-level geocode with its parent street segment. This provides you with additional data about the parent street segment that is not retrievable using only the point-level match. The output information also includes the bearing from the point data geocode to the centerline match.

Street Matching

Street matching identifies the approximate location of an address on a street segment. In street matching, the location is determined by calculating the approximate location of a house number based on the range of numbers in the location's street. For example, if the address is on a street segment with a range of addresses from 50 to 99, then it is assumed that the house number 75 would be in the middle of the street segment. This method assumes that the addresses are evenly spaced along the street segment. As a result, it is not as exact as point matching since addresses may not be evenly distributed along a street segment.

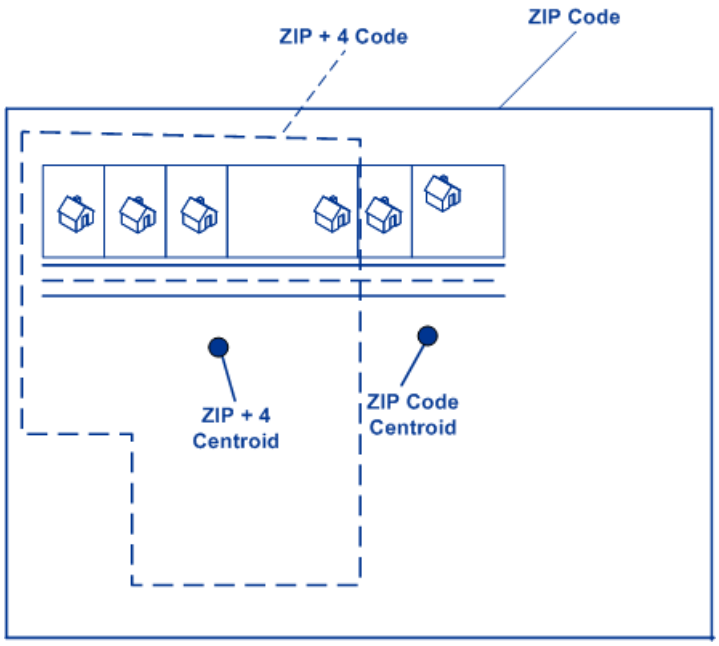
For example, the following diagram shows the results of street-level matching along a segment with unevenly-spaced buildings. The first three buildings are fairly accurately geocoded because they are evenly spaced. The fourth building, however, resides on a slightly larger parcel than the others along this street. Since street-level matching assumes that the buildings are evenly spaced, the result is that fourth, fifth, and sixth houses are not as precise as the first three. If you were to use point-level geocoding, the results would be more accurate.



Centroid Matching

ZIP Code centroid matching is a center point of an area defined by either a ZIP Code or a ZIP + 4, and is the least accurate type of geocode. A ZIP Centroid is the center of a ZIP Code; a ZIP + 4 centroid is the center of a ZIP + 4. Since a ZIP + 4 represents a smaller area than a ZIP Code, a ZIP + 4 centroid is more accurate than a ZIP Code centroid.

The following diagram illustrates centroid matching. All six houses would have the same geocode in this example because they all reside in the same ZIP + 4 code.



Geocoding Match Strategies for Non-U.S. Locations

The Enterprise Geocoding Module offers a variety of options for controlling geocoding precision and match rate. The following information describes different approaches for matching which you can apply to any country geocoder except the U.S. geocoder (GeocodeUSAddress), which has a different set of options.

Maximizing the Match Rate

To generate the highest match rate possible, do not specify house number, street, and city/locality using the ExactMatch option.

Another way to maximize the match rate is by setting FallbackToPostal=Y. This means that the geocoder will fall back to the four-digit postcode centroid if a close street level match cannot be made. While this scenario might yield false positives, it may be the best matching solution when you have large databases to geocode.

You should evaluate if the percentage of false positives will affect your analysis. To reduce the number of false positives without sacrificing hit rate, analyze the result codes after a geocoding session and adjust your settings accordingly.

Maximizing Precision

If your analysis requires highly precise geocoded addresses, choose a strategy in which the geocoder returns the maximum percentage of high precision geocodes and the lowest number of imprecise matches (false positives). To do this, use the ExactMatch option to require close matches to match on all address elements. Also, set FallbackToPostal=N.

This technique may produce a lower percentage match rate, but will provide the best precision.

Balancing Match Rate and Precision

You may want to use a balanced strategy between match rate and geographic precision. That is, you may want to geocode as many records as possible automatically, but at the same time want to minimize the number of weaker matches (false positives). For example, false positives can occur when the geocoder:

- finds a street that sounds like the input street.

- finds the same street in another city (if postal code match is not required).
- finds the street but with a different house number (if house number is not required).

The following settings may achieve a good balance between match rate and precision:

- **CloseMatchesOnly**—Specify "Y".
- **MustMatchHouseNumber**—Specify "Y".
- **MustMatchStreet**—Specify "Y".
- **FallbackToPostal**—Specify "N".

Postal Concepts

The following sections contain information on postal concepts used by the Enterprise Geocoding Module.

Dual Addresses

GeocodeUSAddress can process input that contains two addresses for the same record on the same address line. For example, GeocodeUSAddress can process the following input address:

3138 HWY 371
PO BOX 120
PRESCOTT AR 71857

GeocodeUSAddress does not recognize dual addresses where the two addresses are both street addresses. For example, GeocodeUSAddress does NOT recognize 135 Main St 4750 Walnut St Ste 200. GeocodeUSAddress does recognize dual addresses where the two addresses are the same type of address but are not street addresses. For example, GeocodeUSAddress does recognize PO BOX 12 PO BOX 2000.

After GeocodeUSAddress parses the dual address, it searches for a match. GeocodeUSAddress determines which address has preference for a match based on the processing mode. In CASS mode, GeocodeUSAddress ignores the prefer PO Box and prefer street options, and attempts to find a match based on the following order: PO Box, Street, Rural Route, and General Delivery. In Relaxed mode, GeocodeUSAddress recognizes the Address Preference (AddressPreference) input option.

Note: GeocodeUSAddress does not perform dual address processing in Exact and Close mode. GeocodeUSAddress does not perform dual address processing on multi-line addresses.

Locatable Address Conversion System (LACS)

The USPS® Locatable Address Conversion System (LACS) corrects addresses that have changed as a result of a rural route address converting to street-style address, a PO Box renumbering, or a street-style address changing. The following are examples of LACS^{Link} conversions:

- Rural Route Converted to Street-Style Address: Old Address: RR 3 Box 45 New Address: 1292 North Ridgeland Drive
- Street Renamed and Renumbered: Old Address: 23 Main Street New Address: 45 West First Avenue
- PO Box Renumbered: Old Address: PO Box 453 New Address: PO Box 10435

LACS^{Link} is required for CASS processing.

Delivery Point Validation (DPV)

Delivery Point Validation (DPV®) is a United States Postal Service® (USPS®) technology that validates the accuracy of address information down to the individual mailing address. By using DPV® to validate addresses, you can reduce undeliverable-as-addressed (UAA) mail, thereby reducing postage costs and other business costs associated with inaccurate address information.

Note: DPV® is only available for U.S. addresses.

Without DPV®, the address validation process only verifies that an individual address is within a range of valid addresses for the given street. For example, the USPS data indicates that the range of addresses on Maple Lane is 500 to 1000. You attempt to validate an address of 610 Maple Ln. Without DPV®, this address would appear to be valid because it is in the range of 500 to 1000. However, in reality the address 610 Maple Ln does not exist: the house numbers in this section of the street are 608, 609, 613, and 616. With DPV® processing, you would be alerted to the fact that 610 Maple Ln does not exist and you could take action to correct the address.

DPV® also provides unique address attributes to help produce more targeted mailing lists. For example, DPV® can indicate if a location is vacant and can identify commercial mail receiving agencies (CMRAs) and private mail boxes.

Although DPV® can validate the accuracy of an existing address, you cannot use DPV® to create address lists. For example, you can validate that 123 Elm Street Apartment 6 exists, but you cannot ask if there is an Apartment 7 at the same street address. To prevent the generation of address lists, the DPV® database contains false positive records. False positive records are artificially manufactured addresses that reside in a false positive table. For each negative response that occurs in a DPV® query, a query is made to the false positive table. A match to this table will stop DPV® processing.

Early Warning System (EWS)

The Early Warning System (EWS) provides up-to-date address information for new and recently changed addresses that have not yet been updated in the monthly USPS database. EWS prevents address records from miscoding due to a delay in postal data reaching the USPS® databases.

The older the U.S. Postal Database, the higher potential you have for miscoding addresses. When a valid address is miscoded because the address it matches to in the U.S. Postal Database is inexact, it will result in a broken address.

EWS data consists of partial address information limited to the ZIP Code™, street name, predirectional, postdirectional, and a suffix. For an address record to be EWS-eligible, it must be an address not present on the most recent monthly production U.S. Postal Database.

The USPS® refreshes the EWS file on a weekly basis. You can download the EWS file from the USPS® website at **https://ribbs.usps.gov/cassmass/documents/tech_guides/**.

Geocode Address Global

For information on using the API to access Geocode Address Global, see the geocoding guides.

GeocodeAddressWorld

The GeocodeAddressWorld component takes an address located in any of the supported countries and returns the city centroid or, for some countries, postal centroid. The GeocodeAddressWorld component cannot geocode to the street address level. If you require address-level geocoding, use GeocodeAddressGlobal.

GeocodeAddressWorld is typically used as a fallback geocoder to cover countries for which a Geocode Address Global country is not available. For example, you may have licensed the Australia geocoder because you are primarily interested in geocoding Australian addresses. However, your data may have some records with locations outside Australia. In this case you could use GeocodeAddressWorld to provide centroid geocodes for locations outside Australia, while using the Australia geocoders to provide more precise geocodes for Australian addresses. In other dataflows, you may choose to use GeocodeAddressWorld as a first pass geocoder and then route the results to country-specific geocoders. The best strategy depends on your business case and the nature of your address data.

The GeocodeAddressWorld component is an optional part of the Enterprise Geocoding Module. For more information about Enterprise Geocoding Module, including a listing of other components included with it, see **Enterprise Geocoding Module** on page 204.

Geocode Precision

GeocodeAddressWorld automatically provides the best geocode possible based on the data you provide on input. If you provide a city and valid postal code, you will receive a postal code centroid. If you provide a city and an invalid postal code, or a city and no postal code, GeocodeAddressWorld will return the geographic centroid of the city.

See **Geographic Geocoding** on page 214 and **Postal Geocoding** on page 213.

From Management Console, you can select Geographic or Postal geocoding. You can also select Best Match. In both geographic and postal geocoding are possible, the Best Match selection will return a close match geographic candidate if the geographic result is to a city level or better (that is, a G3 or G4 result code). If the geographic result is less accurate than a city level (that is, a G1 or G2 result code), then Best Match may return a postal (Z1 result). If a postal result is not available, then the best available geographic candidate is returned.

See **Geographic Geocoding Result Codes** and **Postal Geocoding Result Codes**.

Postal Geocoding

Geocode Address World can geocode to a postal centroid if postcode information is available from the country. Postcode information can come from any of the data sources (TomTom, GeoNames, or Pitney Bowes). See **Country Postal Data Coverage** on page 225 for a summary of Geocode Address World postal data coverage. Depending on the country, postal geocoding may provide more accurate results that geographic geocoding.

Postal level geocoding is possible if these conditions are met:

- Your input address consists of a valid postcode.
- The data source contains postcode information for the country. Not every country has postcode data.

Geocode Address World may return multiple close matches for postal geocoding. For example, a postcode of 12180 matches Troy NY but the identical postcode occurs in several other countries. If the input is the postcode only, then all those candidates are returned as close matches.

However, if the input includes geographic address elements (such as country, state, region, or city name), Geocode Address World may be able to use that information to return a more accurate single close match. If you want to use geographic address content to refine your postal geocoding results, consider the following:

- Different countries derive their postal data from either the TomTom, GeoNames, or Pitney Bowes sources. Therefore, the available geographic content in the postal data source varies by country. For example, city name (City) is a close match weighting factor for countries that use the GeoNames postal data source, but city name is ignored for countries that use the TomTom postal data source. See **Data Sources and Coverage** on page 216 for information on the geographic content of the TomTom, GeoNames, and Pitney Bowes data sources.

Postal Geocoding with Geographic Information

In this postal geocoding example, the input address includes a valid postcode of 41012 and the province (StateProvince) of Emilia Romagna. A street address is provided, but this is ignored for postal geocoding.

Fornaci 40
Emilia Romagna
41012

Because the TomTom postal data source for Italy includes StateProvince, the province of Emilia Romagna is considered when evaluating close matches. Therefore, Emilia Romagna, Italy with the matching 41012 postal code is returned as the single close match with a Z1 result code. Candidates with a 41012 postcode from other countries are returned as non-close candidates. If StateProvince or Country information was not provided on input, then Geocode Address World would return multiple close matches because the five-digit 41012 postcode can be found in a number of countries.

The geographic content must be present in the postal data source in order refine postal geocoding results. For example, the Italy TomTom postal data source does not include city/town (City). So if you input the city of Carpi with the 41012 postal code, Geocode Address World ignores the city name and returns multiple close matches for the 41012 postal code (unless you also specified the ITA country name). See **Data Sources and Coverage** on page 216 for information on the geographic content of the TomTom, GeoNames, and Pitney Bowes data sources.

Geographic Geocoding

Geocode Address World can geocode to the centroid of an administrative division (such as town or village).

Geocode Address World can geocode to the geographic level if these conditions are met:

- Your input addresses contains accurate geographic information without valid postcode address content in the input. If the address in question includes valid postcode input, then Geocode Address World will attempt postal geocoding.
- The data source contains geographic level information for the country. Geographic information can come from any of the data sources (TomTom, GeoNames, or Pitney Bowes).
- Country name or country ISO codes are not required, but if included, they must be matched. Including the country name may produce better close matches.

Geographic Geocoding to City

In this example, the input address includes the city (City) of Vaihingen an der Enz. The country is not specified in this example. The street address information (street name and number) is ignored for the purposed of geographic geocoding.

Muldenweg 2
Vaihingen an der Enz

Geocode Address World returns a G3 close match candidate. Even though the country was not specified, Geocode Address World identifies one close match in Germany (DEU).

StateProvince: Baden-Württemberg
County: Ludwigsburg
City: Vaihingen an der Enz
Country: DEU
Result Code: G3
X: 8.95948
Y: 48.930059

Geographic Geocoding with Common City Name

In this example, the input address includes the city (City) of Venice. This city name occurs in a number of countries, but the country is not specified on input.

St Marks Plaza
Venice

Geocode Address World selects Venice, Italy as the close match candidate because of its large population (approximately 270,000) and because Venice is the administrative capital of the Veneto region of Italy. A number of non-close matches may also be returned for cities of Venice in other countries. The close match candidate for Venice, ITA is:

StateProvince: Veneto
County: Venezia
City: Venice
Country: ITA
Result Code: G3
X: 12.33878
Y: 45.43434

Geographic Geocoding with State/Province Abbreviation

In this example the input address includes the city name of Rome and GA, which is the abbreviation for the state of Georgia in the USA. See **State or Province Abbreviations** on page 233 to see the countries for which state/province abbreviations are recognized. Because the state abbreviation is used, it is not necessary to specify the country name.

Rome, GA

Geocode Address World considers the StateProvince and returns a close match for Rome, Georgia USA. Even though Rome, Italy is a much larger city and is the capital of Italy, that is returned as a non-close candidate because the StateProvince (GA) that was specified on input

StateProvince: Georgia
County: Floyd
City: Rome
Country: USA
Result Code: G3
X: -85.16467
Y: 34.25704

Geographic Geocoding to Locality

In this example, the input address includes the locality of Altamira and province abbreviation of GRO Geocode Address World recognizes the state abbreviation of GRO, so the country name is not necessary.

City: Altamira
StateProvince: GRO

In this example, Geocode Address World returns a close match to Locality) of Altamira even if Altamira was input as City. The (StateProvince) of GRO is also returned. If Guerrero is entered as StateProvince then Guerrero is returned.

StateProvince: GRO
City: ACAPULCO DE JUÁREZ
Locality: ALTAMIRA
Country: MEX
Result Code: G4
X: 99.87984
Y: 16.87637

Address input can be formatted into separate input fields or input can be unformatted (single line input). Geocoding of unformatted input is shown in **Single Line Input** on page 235.

Geographic Areas

Every country has administrative divisions and many of these administrative areas are used in addresses. Geocode Address World identifies four AreaNames, each one corresponding to an administrative division. Administrative division naming and hierarchy vary by country.

- locality
- city
- county
- state/province

.

Data Sources and Coverage

Geocode Address World relies on several data sources to build its comprehensive worldwide address databases. If an input address cannot be located using one of these data sources, then Geocode Address World uses one of the other data sources. The best available candidate is returned.

These data sources (for both geographic and postal data) are used in the listed order:

- TomTom data
- GeoNames data
- Pitney Bowes World data

Geocode Address World is partitioned into six databases based on continent. The geographic and postal data is integrated into each address dictionary to support both geographic geocoding and postal geocoding.

- Africa
- Asia
- Europe
- NorthAmerica
- Oceana
- SouthAmerica

See **Geographic Geocoding** on page 214 for a description and examples of geographic geocoding. See **Postal Geocoding** on page 213 for a description and examples of postal geocoding.

The postal source data can access the geographic content, which can be used to refine postal results. That is, geographic information (country name and administrative divisions) can be used to help evaluate close matches when the same postal code can be found in different countries.

Depending on the source of the postal data, the following geographic information is available to help refine postal results:

- TomTom source: Country, StateProvince
- GeoNames source: Country, StateProvince and City
- Pitney Bowes World source: Country, StateProvince, County, City, and Locality

Note: The Geocode Address World data set contains data licensed from the GeoNames Project (**http://www.geonames.org**) provided under the Creative Commons Attribution License ("Attribution License") located at **http://creativecommons.org/licenses/by/3.0/legalcode**. Your use of the GeoNames data (described in the Spectrum User Manual) is governed by the terms of the Attribution License, and any conflict between your agreement with PBSI and the Attribution License will be resolved in favor of the Attribution License solely as it relates to your use of the GeoNames data.

Country Coverage

Geocode Address World includes coverage for almost every country in the world. The accuracy and scope of coverage varies depending on the quality of the available data source. Some countries include postcode data, while other countries have geographic coverage only.

See **Data Sources and Coverage** on page 216 for more information on the TomTom, GeoNames, and Pitney Bowes geographic and postal data sources.

For a complete list of Geographic coverage by country, see **Country Geographic Data Coverage** on page 217. For Postal coverage by country, see **Country Postal Data Coverage** on page 225.

Country Geographic Data Coverage

Table 42: Country Names and Geographic Data Coverage

Country Name	ISO 3166 Country Code	Data Source	Vintage
AFGHANISTAN	AFG	GeoNames	2011.07
ALAND ISLANDS	ALA	GeoNames	2011.07
ALBANIA	ALB	TomTom	2011.06
ALGERIA	DZA	GeoNames	2011.07
AMERICAN SAMOA	ASM	GeoNames	2011.07
ANDORRA	AND	TomTom	2011.06
ANGOLA	AGO	TomTom	2011.06
ANGUILLA	AIA	GeoNames	2011.07
ANTARCTICA	ATA	GeoNames	2011.07
ANTIGUA AND BARBUDA	ATG	GeoNames	2011.07
ARGENTINA	ARG	TomTom	2011.06
ARMENIA	ARM	GeoNames	2011.07
ARUBA	ABW	GeoNames	2011.07
AUSTRALIA	AUS	GeoNames	2011.07
AUSTRIA	AUT	TomTom	2011.06
AZERBAIJAN	AZE	GeoNames	2011.07
BAHAMAS	BHS	GeoNames	2011.07
BAHRAIN	BHR	TomTom	2011.06
BANGLADESH	BGD	GeoNames	2011.07
BARBADOS	BRB	GeoNames	2011.07
BELARUS	BLR	TomTom	2011.06
BELGIUM	BEL	TomTom	2011.06
BELIZE	BLZ	GeoNames	2011.07
BENIN	BEN	TomTom	2011.06
BERMUDA	BMU	GeoNames	2011.07
BHUTAN	BTN	GeoNames	2011.07
BOLIVIA	BOL	GeoNames	2011.07

Country Name	ISO 3166 Country Code	Data Source	Vintage
BONAIRE, SINT EUSTATIUS AND SABA	BES	GeoNames	2011.07
BOSNIA AND HERZEGOWINA	BIH	TomTom	2011.06
BOTSWANA	BWA	TomTom	2011.06
BOUVET ISLAND	BVT	GeoNames	2011.07
BRAZIL	BRA	TomTom	2011.06
BRITISH INDIAN OCEAN TERRITORY	IOT	GeoNames	2011.07
BRUNEI DARUSSALAM	BRN	TomTom	2011.06
BULGARIA	BGR	TomTom	2011.06
BURKINA FASO	BFA	TomTom	2011.06
BURUNDI	BDI	GeoNames	2011.07
CAMBODIA	KHM	GeoNames	2011.07
CAMEROON	CMR	TomTom	2011.06
CANADA	CAN	TomTom	2011.06
CAPE VERDE	CPV	GeoNames	2011.07
CAYMAN ISLANDS	CYM	GeoNames	2011.07
CENTRAL AFRICAN REPUBLIC	CAF	GeoNames	2011.07
CHAD	TCD	GeoNames	2011.07
CHILE	CHL	TomTom	2011.06
CHINA	CHN	GeoNames	2011.07
CHRISTMAS ISLAND	CXR	GeoNames	2011.07
COCOS (KEELING) ISLANDS	CCK	GeoNames	2011.07
COLOMBIA	COL	GeoNames	2011.07
COMOROS	COM	GeoNames	2011.07
CONGO	COG	TomTom	2011.06
CONGO, DEMOCRATIC REPUBLIC OF THE	COD	TomTom	2011.06
COOK ISLANDS	COK	GeoNames	2011.07
COSTA RICA	CRI	GeoNames	2011.07
COTE D'IVOIRE	CIV	GeoNames	2011.07
CROATIA (LOCAL NAME: HRVATSKA)	HRV	TomTom	2011.06
CUBA	CUB	GeoNames	2011.07

Country Name	ISO 3166 Country Code	Data Source	Vintage
CURAÇAO	CUW	GeoNames	2011.07
CYPRUS	CYP	GeoNames	2011.07
CZECH REPUBLIC	CZE	TomTom	2011.06
DENMARK	DNK	GeoNames	2011.07
DJIBOUTI	DJI	GeoNames	2011.07
DOMINICA	DMA	GeoNames	2011.07
DOMINICAN REPUBLIC	DOM	GeoNames	2011.07
ECUADOR	ECU	GeoNames	2011.07
EGYPT	EGY	TomTom	2011.06
EL SALVADOR	SLV	GeoNames	2011.07
EQUATORIAL GUINEA	GNQ	GeoNames	2011.07
ERITREA	ERI	GeoNames	2011.07
ESTONIA	EST	TomTom	2011.06
ETHIOPIA	ETH	GeoNames	2011.07
FALKLAND ISLANDS (MALVINAS)	FLK	GeoNames	2011.07
FAROE ISLANDS	FRO	GeoNames	2011.07
FIJI	FJI	GeoNames	2011.07
FINLAND	FIN	TomTom	2011.06
FRANCE	FRA	TomTom	2011.06
FRENCH GUIANA	GUF	TomTom	2011.06
FRENCH POLYNESIA	PYF	GeoNames	2011.07
FRENCH SOUTHERN TERRITORIES	ATF	GeoNames	2011.07
GABON	GAB	TomTom	2011.06
GAMBIA	GMB	GeoNames	2011.07
GEORGIA	GEO	GeoNames	2011.07
GERMANY	DEU	TomTom	2011.06
GHANA	GHA	TomTom	2011.06
GIBRALTAR	GIB	GeoNames	2011.07
GREECE	GRC	TomTom	2011.06
GREENLAND	GRL	GeoNames	2011.07
GRENADA	GRD	GeoNames	2011.07

Country Name	ISO 3166 Country Code	Data Source	Vintage
GUADELOUPE	GLP	TomTom	2011.06
GUAM	GUM	GeoNames	2011.07
GUATEMALA	GTM	GeoNames	2011.07
GUERNSEY	GGY	GeoNames	2011.07
GUINEA	GIN	GeoNames	2011.07
GUINEA-BISSAU	GNB	GeoNames	2011.07
GUYANA	GUY	GeoNames	2011.07
HAITI	HTI	GeoNames	2011.07
HEARD AND MCDONALD ISLANDS	HMD	GeoNames	2011.07
HONDURAS	HND	GeoNames	2011.07
HONG KONG	HKG	TomTom	2011.06
HUNGARY	HUN	TomTom	2011.06
ICELAND	ISL	GeoNames	2011.07
INDIA	IND	GeoNames	2011.07
INDONESIA	IDN	TomTom	2011.06
IRAN (ISLAMIC REPUBLIC OF)	IRN	GeoNames	2011.07
IRAQ	IRQ	GeoNames	2011.07
IRELAND	IRL	TomTom	2011.06
ISLE OF MAN	IMN	GeoNames	2011.07
ISRAEL	ISR	GeoNames	2011.07
ITALY	ITA	TomTom	2011.06
JAMAICA	JAM	GeoNames	2011.07
JAPAN	JPN	GeoNames	2011.07
JERSEY	JEY	GeoNames	2011.07
JORDAN	JOR	GeoNames	2011.07
KAZAKHSTAN	KAZ	GeoNames	2011.07
KENYA	KEN	TomTom	2011.06
KIRIBATI	KIR	GeoNames	2011.07
KOREA, DEMOCRATIC PEOPLE'S REPUBLIC OF	PRK	GeoNames	2011.07
KOREA, REPUBLIC OF	KOR	GeoNames	2011.07
KUWAIT	KWT	TomTom	2011.06

Country Name	ISO 3166 Country Code	Data Source	Vintage
KYRGYZSTAN	KGZ	GeoNames	2011.07
LAO PEOPLE'S DEMOCRATIC REPUBLIC	LAO	GeoNames	2011.07
LATVIA	LVA	TomTom	2011.06
LEBANON	LBN	GeoNames	2011.07
LESOTHO	LSO	TomTom	2011.06
LIBERIA	LBR	GeoNames	2011.07
LIBYAN ARAB JAMAHIRIYA	LBY	GeoNames	2011.07
LIECHTENSTEIN	LIE	GeoNames	2011.07
LITHUANIA	LTU	TomTom	2011.06
LUXEMBOURG	LUX	TomTom	2011.06
MACAO	MAC	TomTom	2011.06
MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MKD	TomTom	2011.06
MADAGASCAR	MDG	GeoNames	2011.07
MALAWI	MWI	TomTom	2011.06
MALAYSIA	MYS	TomTom	2011.06
MALDIVES	MDV	GeoNames	2011.07
MALI	MLI	TomTom	2011.06
MALTA	MLT	TomTom	2011.06
MARSHALL ISLANDS	MHL	GeoNames	2011.07
MARTINIQUE	MTQ	GeoNames	2011.07
MAURITANIA	MRT	TomTom	2011.06
MAURITIUS	MUS	TomTom	2011.06
MAYOTTE	MYT	GeoNames	2011.07
MEXICO	MEX	TomTom	2011.06
MICRONESIA, FEDERATED STATES OF	FSM	GeoNames	2011.07
MOLDOVA, REPUBLIC OF	MDA	TomTom	2011.06
MONACO	MCO	GeoNames	2011.07
MONGOLIA	MNG	GeoNames	2011.07
MONTENEGRO	MNE	TomTom	2011.06
MONTSERRAT	MSR	GeoNames	2011.07

Country Name	ISO 3166 Country Code	Data Source	Vintage
MOROCCO	MAR	TomTom	2011.06
MOZAMBIQUE	MOZ	TomTom	2011.06
MYANMAR	MMR	GeoNames	2011.07
NAMIBIA	NAM	GeoNames	2011.07
NAURU	NRU	GeoNames	2011.07
NEPAL	NPL	GeoNames	2011.07
NETHERLANDS	NLD	TomTom	2011.06
NETHERLANDS ANTILLES	ANT	Pitney Bowes	C. 2006
NEW CALEDONIA	NCL	GeoNames	2011.07
NEW ZEALAND	NZL	GeoNames	2011.07
NICARAGUA	NIC	GeoNames	2011.07
NIGER	NER	TomTom	2011.06
NIGERIA	NGA	TomTom	2011.06
NIUE	NIU	GeoNames	2011.07
NORFOLK ISLAND	NFK	GeoNames	2011.07
NORTHERN MARIANA ISLANDS	MNP	GeoNames	2011.07
NORWAY	NOR	TomTom	2011.06
OMAN	OMN	TomTom	2011.06
PAKISTAN	PAK	GeoNames	2011.07
PALAU	PLW	GeoNames	2011.07
PALESTINIAN TERRITORY, OCCUPIED	PSE	GeoNames	2011.07
PANAMA	PAN	GeoNames	2011.07
PAPUA NEW GUINEA	PNG	GeoNames	2011.07
PARAGUAY	PRY	GeoNames	2011.07
PERU	PER	GeoNames	2011.07
PHILIPPINES	PHL	TomTom	2011.06
PITCAIRN	PCN	GeoNames	2011.07
POLAND	POL	TomTom	2011.06
PORTUGAL	PRT	TomTom	2011.06
PUERTO RICO	PRI	GeoNames	2011.07
QATAR	QAT	TomTom	2011.06

Country Name	ISO 3166 Country Code	Data Source	Vintage
REUNION	REU	TomTom	2011.06
ROMANIA	ROU	TomTom	2011.06
RUSSIAN FEDERATION	RUS	TomTom	2011.06
RWANDA	RWA	GeoNames	2011.07
SAINT BARTHÉLEMY	BLM	GeoNames	2011.07
SAINT HELENA, ASCENSION AND TRISTAN DA CUNHA	SHN	GeoNames	2011.07
SAINT KITTS AND NEVIS	KNA	GeoNames	2011.07
SAINT LUCIA	LCA	GeoNames	2011.07
SAINT MARTIN (FRENCH PART	MAF	GeoNames	2011.07
SAINT PIERRE AND MIQUELON	SPM	GeoNames	2011.07
SAINT VINCENT AND THE GRENADINES	VCT	GeoNames	2011.07
SAMOA	WSM	GeoNames	2011.07
SAN MARINO	SMR	TomTom	2011.06
SAO TOME AND PRINCIPE	STP	GeoNames	2011.07
SAUDI ARABIA	SAU	TomTom	2011.06
SENEGAL	SEN	TomTom	2011.06
SERBIA	SRB	TomTom	2011.06
SEYCHELLES	SYC	GeoNames	2011.07
SIERRA LEONE	SLE	GeoNames	2011.07
SINGAPORE	SGP	TomTom	2011.06
SINT MAARTEN (DUTCH PART)	SXM	GeoNames	2011.07
SLOVAKIA (SLOVAK REPUBLIC)	SVK	TomTom	2011.06
SLOVENIA	SVN	TomTom	2011.06
SOLOMON ISLANDS	SLB	GeoNames	2011.07
SOMALIA	SOM	GeoNames	2011.07
SOUTH AFRICA	ZAF	GeoNames	2011.07
SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	SGS	GeoNames	2011.07
SPAIN	ESP	TomTom	2011.06
SRI LANKA	LKA	GeoNames	2011.07
SUDAN	SDN	GeoNames	2011.07

Country Name	ISO 3166 Country Code	Data Source	Vintage
SURINAME	SUR	GeoNames	2011.07
SVALBARD AND JAN MAYEN ISLANDS	SJM	GeoNames	2011.07
SWAZILAND	SWZ	TomTom	2011.06
SWEDEN	SWE	TomTom	2011.06
SWITZERLAND	CHE	TomTom	2011.06
SYRIAN ARAB REPUBLIC	SYR	GeoNames	2011.07
TAIWAN	TWN	TomTom	2011.06
TAJIKISTAN	TJK	GeoNames	2011.07
TANZANIA, UNITED REPUBLIC OF	TZA	TomTom	2011.06
THAILAND	THA	TomTom	2011.06
TIMOR-LESTE	TLS	GeoNames	2011.07
TOGO	TGO	TomTom	2011.06
TOKELAU	TKL	GeoNames	2011.07
TONGA	TON	GeoNames	2011.07
TRINIDAD AND TOBAGO	TTO	GeoNames	2011.07
TUNISIA	TUN	GeoNames	2011.07
TURKEY	TUR	TomTom	2011.06
TURKMENISTAN	TKM	GeoNames	2011.07
TURKS AND CAICOS ISLANDS	TCA	GeoNames	2011.07
TUVALU	TUV	GeoNames	2011.07
UGANDA	UGA	TomTom	2011.06
UKRAINE	UKR	TomTom	2011.06
UNITED ARAB EMIRATES	ARE	TomTom	2011.06
UNITED KINGDOM	GBR	TomTom	2011.06
UNITED STATES	USA	GeoNames	2011.07
UNITED STATES MINOR OUTLYING ISLANDS	UMI	GeoNames	2011.07
URUGUAY	URY	TomTom	2011.06
UZBEKISTAN	UZB	GeoNames	2011.07
VANUATU	VUT	GeoNames	2011.07
VATICAN CITY STATE (HOLY SEE)	VAT	GeoNames	2011.07
VENEZUELA	VEN	GeoNames	2011.07

Country Name	ISO 3166 Country Code	Data Source	Vintage
VIET NAM	VNM	GeoNames	2011.07
VIRGIN ISLANDS (BRITISH)	VGB	GeoNames	2011.07
VIRGIN ISLANDS (U.S.)	VIR	GeoNames	2011.07
WALLIS AND FUTUNA ISLANDS	WLF	GeoNames	2011.07
WESTERN SAHARA	ESH	GeoNames	2011.07
YEMEN	YEM	GeoNames	2011.07
ZAMBIA	ZMB	TomTom	2011.06
ZIMBABWE	ZWE	GeoNames	2011.07

Country Postal Data Coverage

Table 43: Country Names and Postal Data Coverage

Country Name	ISO 3166 Country Code	Data Source	Vintage
ALGERIA	DZA	Pitney Bowes	C. 2006
AMERICAN SAMOA	ASM	GeoNames	2011.07
ANDORRA	AND	TomTom	2011.06
ARGENTINA	ARG	GeoNames	2011.07
ARMENIA	ARM	Pitney Bowes	C. 2006
AUSTRALIA	AUS	GeoNames	2011.07
AUSTRIA	AUT	TomTom	2011.06
AZERBAIJAN	AZE	Pitney Bowes	C. 2006
BAHRAIN	BHR	Pitney Bowes	C. 2006
BANGLADESH	BGD	GeoNames	2011.07
BELARUS	BLR	Pitney Bowes	C. 2006
BELGIUM	BEL	TomTom	2011.06
BERMUDA	BMU	Pitney Bowes	C. 2006
BOSNIA AND HERZEGOWINA	BIH	Pitney Bowes	C. 2006
BRAZIL	BRA	TomTom	2011.09
BRITISH INDIAN OCEAN TERRITORY	IOT	Pitney Bowes	C. 2006
BRUNEI DARUSSALAM	BRN	Pitney Bowes	C. 2006
BULGARIA	BGR	GeoNames	2011.07
CAMBODIA	KHM	Pitney Bowes	C. 2006

Country Name	ISO 3166 Country Code	Data Source	Vintage
CANADA	CAN	TomTom	2011.09
CAPE VERDE	CPV	Pitney Bowes	C. 2006
CHILE	CHL	Pitney Bowes	C. 2006
CHINA	CHN	Pitney Bowes	C. 2006
CHRISTMAS ISLAND	CXR	Pitney Bowes	C. 2006
COCOS (KEELING) ISLANDS	CCK	Pitney Bowes	C. 2006
COSTA RICA	CRI	Pitney Bowes	C. 2006
CROATIA (LOCAL NAME: HRVATSKA)	HRV	GeoNames	2011.07
CUBA	CUB	Pitney Bowes	C. 2006
CYPRUS	CYP	Pitney Bowes	C. 2006
CZECH REPUBLIC	CZE	TomTom	2011.06
DENMARK	DNK	GeoNames	2011.07
DOMINICAN REPUBLIC	DOM	GeoNames	2011.07
ECUADOR	ECU	Pitney Bowes	C. 2006
EGYPT	EGY	Pitney Bowes	C. 2006
EL SALVADOR	SLV	Pitney Bowes	C. 2006
ESTONIA	EST	TomTom	2011.06
ETHIOPIA	ETH	Pitney Bowes	C. 2006
FALKLAND ISLANDS (MALVINAS)	FLK	Pitney Bowes	C. 2006
FAROE ISLANDS	FRO	GeoNames	2011.07
FINLAND	FIN	TomTom	2011.06
FRANCE	FRA	TomTom	2011.06
FRENCH GUIANA	GUF	GeoNames	2011.07
FRENCH POLYNESIA	PYF	Pitney Bowes	C. 2006
GEORGIA	GEO	Pitney Bowes	C. 2006
GERMANY	DEU	TomTom	2011.06
GREECE	GRC	TomTom	2011.06
GREENLAND	GRL	GeoNames	2011.07
GUADELOUPE	GLP	GeoNames	2011.07
GUAM	GUM	GeoNames	2011.07
GUATEMALA	GTM	GeoNames	2011.07

Country Name	ISO 3166 Country Code	Data Source	Vintage
GUERNSEY	GGY	GeoNames	2011.07
GUINEA	GIN	Pitney Bowes	C. 2006
GUINEA-BISSAU	GNB	Pitney Bowes	C. 2006
HAITI	HTI	Pitney Bowes	C. 2006
HONDURAS	HND	Pitney Bowes	C. 2006
HUNGARY	HUN	GeoNames	2011.07
ICELAND	ISL	GeoNames	2011.07
INDIA	IND	GeoNames	2011.07
INDONESIA	IDN	TomTom	2011.06
IRAN (ISLAMIC REPUBLIC OF)	IRN	Pitney Bowes	C. 2006
IRAQ	IRQ	Pitney Bowes	C. 2006
IRELAND	IRL	Pitney Bowes	C. 2006
ISLE OF MAN	IMN	GeoNames	2011.07
ISRAEL	ISR	Pitney Bowes	C. 2006
ITALY	ITA	TomTom	2011.06
JAMAICA	JAM	Pitney Bowes	C. 2006
JAPAN	JPN	GeoNames	2011.07
JERSEY	JEY	GeoNames	2011.07
JORDAN	JOR	Pitney Bowes	C. 2006
KAZAKHSTAN	KAZ	Pitney Bowes	C. 2006
KENYA	KEN	Pitney Bowes	C. 2006
KOREA, REPUBLIC OF	KOR	Pitney Bowes	C. 2006
KUWAIT	KWT	Pitney Bowes	C. 2006
KYRGYZSTAN	KGZ	Pitney Bowes	C. 2006
LAO PEOPLE'S DEMOCRATIC REPUBLIC	LAO	Pitney Bowes	C. 2006
LATVIA	LVA	TomTom	2011.06
LEBANON	LBN	Pitney Bowes	C. 2006
LESOTHO	LSO	Pitney Bowes	C. 2006
LIBERIA	LBR	Pitney Bowes	C. 2006
LIECHTENSTEIN	LIE	GeoNames	2011.07
LITHUANIA	LTU	TomTom	2011.06

Country Name	ISO 3166 Country Code	Data Source	Vintage
LUXEMBOURG	LUX	GeoNames	2011.07
MACEDONIA, THE FORMER YUGOSLAV REPUBLIC OF	MKD	GeoNames	2011.07
MADAGASCAR	MDG	Pitney Bowes	C. 2006
MALAYSIA	MYS	GeoNames	2011.07
MALDIVES	MDV	Pitney Bowes	C. 2006
MALTA	MLT	Pitney Bowes	C. 2006
MARSHALL ISLANDS	MHL	GeoNames	2011.07
MARTINIQUE	MTQ	GeoNames	2011.07
MAYOTTE	MYT	GeoNames	2011.07
MEXICO	MEX	TomTom	2011.06
MICRONESIA, FEDERATED STATES OF	FSM	Pitney Bowes	C. 2006
MOLDOVA, REPUBLIC OF	MDA	GeoNames	2011.07
MONACO	MCO	GeoNames	2011.07
MONGOLIA	MNG	Pitney Bowes	C. 2006
MOROCCO	MAR	TomTom	2011.06
MOZAMBIQUE	MOZ	Pitney Bowes	C. 2006
MYANMAR	MMR	Pitney Bowes	C. 2006
NEPAL	NPL	Pitney Bowes	C. 2006
NETHERLANDS	NLD	TomTom	2011.06
NEW CALEDONIA	NCL	Pitney Bowes	C. 2006
NEW ZEALAND	NZL	GeoNames	2011.07
NICARAGUA	NIC	Pitney Bowes	C. 2006
NIGER	NER	Pitney Bowes	C. 2006
NIGERIA	NGA	Pitney Bowes	C. 2006
NORFOLK ISLAND	NFK	Pitney Bowes	C. 2006
NORTHERN MARIANA ISLANDS	MNP	GeoNames	2011.07
NORWAY	NOR	TomTom	2011.06
OMAN	OMN	Pitney Bowes	C. 2006
PAKISTAN	PAK	GeoNames	2011.07
PALAU	PLW	Pitney Bowes	C. 2006
PAPUA NEW GUINEA	PNG	Pitney Bowes	C. 2006

Country Name	ISO 3166 Country Code	Data Source	Vintage
PARAGUAY	PRY	Pitney Bowes	C. 2006
PHILIPPINES	PHL	GeoNames	2011.07
PITCAIRN	PCN	Pitney Bowes	C. 2006
POLAND	POL	TomTom	2011.06
PORTUGAL	PRT	TomTom	2011.06
PUERTO RICO	PRI	GeoNames	2011.07
REUNION	REU	GeoNames	2011.07
ROMANIA	ROU	Pitney Bowes	C. 2006
RUSSIAN FEDERATION	RUS	TomTom	2011.06
SAINT HELENA, ASCENSION AND TRISTAN DA CUNHA	SHN	Pitney Bowes	C. 2006
SAINT PIERRE AND MIQUELON	SPM	GeoNames	2011.07
SAN MARINO	SMR	TomTom	2011.06
SAUDI ARABIA	SAU	Pitney Bowes	C. 2006
SENEGAL	SEN	Pitney Bowes	C. 2006
SINGAPORE	SGP	TomTom	2011.06
SLOVAKIA (SLOVAK REPUBLIC)	SVK	TomTom	2011.06
SLOVENIA	SVN	TomTom	2011.06
SOUTH AFRICA	ZAF	GeoNames	2011.07
SOUTH GEORGIA AND THE SOUTH SANDWICH ISLANDS	SGS	Pitney Bowes	C. 2006
SPAIN	ESP	TomTom	2011.06
SRI LANKA	LKA	GeoNames	2011.07
SUDAN	SDN	Pitney Bowes	C. 2006
SWAZILAND	SWZ	Pitney Bowes	C. 2006
SWEDEN	SWE	GeoNames	2011.07
SWITZERLAND	CHE	TomTom	2011.06
TAIWAN	TWN	TomTom	2011.06
TAJIKISTAN	TJK	Pitney Bowes	C. 2006
THAILAND	THA	TomTom	2011.06
TIMOR-LESTE	TLS	Pitney Bowes	C. 2006
TUNISIA	TUN	Pitney Bowes	C. 2006
TURKEY	TUR	TomTom	2011.06

Country Name	ISO 3166 Country Code	Data Source	Vintage
TURKMENISTAN	TKM	Pitney Bowes	C. 2006
TURKS AND CAICOS ISLANDS	TCA	Pitney Bowes	C. 2006
UKRAINE	UKR	Pitney Bowes	C. 2006
UNITED ARAB EMIRATES	ARE	Pitney Bowes	C. 2006
UNITED KINGDOM	GBR	TomTom	2011.06
UNITED STATES	USA	TomTom	2011.06
URUGUAY	URY	Pitney Bowes	C. 2006
UZBEKISTAN	UZB	Pitney Bowes	C. 2006
VATICAN CITY STATE (HOLY SEE)	VAT	TomTom	2011.06
VENEZUELA	VEN	Pitney Bowes	C. 2006
VIET NAM	VNM	Pitney Bowes	C. 2006
VIRGIN ISLANDS (U.S.)	VIR	GeoNames	2011.07
WALLIS AND FUTUNA ISLANDS	WLF	Pitney Bowes	C. 2006
WESTERN SAHARA	ESH	Pitney Bowes	C. 2006
ZAMBIA	ZMB	Pitney Bowes	C. 2006

Spectrum Geocoding Scenarios

You can use Enterprise Manager to create dataflows that are appropriate for your business requirements and for the nature and quality of your data.

Multiple Country Stage with Geocode Address World as Last Geocoding Pass

You may be able to optimize your results by geocoding your input in several passes. In general, you can use more strict matching criteria in the first pass. In subsequent geocoding passes, you can apply less restrictive matching criteria to any addresses that previously failed to return a close match candidate. This strategy can produce accurate matches for your high-quality addresses and still give you the best possible matches for less accurate addresses, or for addresses in countries that do not have a comprehensive level of coverage.

Let's assume the following scenario:

- Your input file includes addresses for six countries: Argentina (ARG), Brazil (BRA), Mexico (MEX), Chile (CHL), Venezuela (VEN), and Panama (PAN).
- You have geocoders for three of these countries (ARG, BRA, and MEX) are deployed in a multiple country stage.
- Geocode Address World is deployed in a separate stage to geocode addresses that could not be identified by the country-specific geocoders.
- Your stage uses conditional routers (and optionally stream combiner) to manage the geocoding flow.

1. Read input into the multiple-country stage. Geocoded addresses can be written out to a file or optionally sent to the stream combiner.

2. Some Addresses that could not be geocoded in step 1. This may be because they were addresses from CHL, VEN, or PAN, and you do not have geocoders for these countries in the first stage. Or they may have failed to return a close match candidate in the first stage because of input errors or ambiguities in the addresses. These ungeocoded addresses are sent to the Geocode Address World stage.
3. Addresses can be geocoded to postal or geographic accuracy by Geocode Address World. Successfully geocoded addresses can be written out to a file or optionally sent to the stream combiner.

   Postal geocoded candidates will have a Z1 result code. Postal geocoded results may be very accurate in countries with robust postcode systems. See **Postal Geocoding** on page 213. Geographic candidates will have a G result code (for example G3 for a town/city match). See **Geographic Geocoding** on page 214.
4. The stream combiner (if used in your dataflow) can combine all geocoded addresses and write them to a file or direct them for further processing.

This is one scenario. You could use Enterprise Manager to design more complex dataflows that are suitable for your needs.

Using Geocode Address World as First Geocoding Pass

You could also use a strategy with Geocode Address World as the first geocoding pass.

Assume the following:

- Your addresses do not typically specify a country (although some may).
- Some addresses contain only street and city address information.
- You have country-specific geocoders for some countries, but not all.
- You use a main dataflow with subflows to manage the geocoding process.

Use a dataflow (possibly with subflows) that perform the following actions. Note that these steps illustrate a simplified view of a sample dataflow.

1. Read input into the mulitple-country stage that also includes Geocode Address World. Based on city name (and possibly state name for USA addresses), each address can produce one or more potential close match candidates for several different countries. Each candidate will now be associated with a country, even though the input address may not have included a country.
2. If a country-specific geocoder is available, the candidate is sent to that geocoder. This processing involves conditional routing, stream combiners, and other Spectrum control stages. Depending on the completeness of the input address and capabilities of the country-specific geocoder, candidates may be geocoded to a street (S result code), geographic (G result code), or postal (Z result code) level.
3. If no country-specific geocoder is available, the candidate is routed to Geocode Address World, where candidates can be geocoded to a geographic or postal level.
4. Candidates from all subflows are combined and ranked using a number of criteria. Ranking could be based on population of the city (city rank), accuracy of the match (street, geographic, postal), proximity to a user's locality, or other criteria.

Input

GeocodeAddressWorld takes an address as input. To obtain the best performance and the most possible matches, your input address lists should be as complete as possible, free of misspellings and incomplete addresses, and as close to postal authority standards as possible. Most postal authorities have websites that contain information about address standards for their particular country.

Note: The country name or two- or three- character country ISO code is optional. If you omit the country, GeocodeAddressWorld returns the best available candidates based on the other input provided.

Input Fields

The following table provides information on the format and layout of GeocodeAddressWorld input.

Note: Specify input using the DataTable class. For information on the DataTable class, see the "API Fundamentals" section.

Table 44: GeocodeAddressWorld Input Data

columnName	Format	Description
AddressLine1	String	The first address line. For example, 4360 DUKES RD: **4360 DUKES RD** KALGOORLIE WA 6430
AddressLine2	String	The second address line of a two-line address. For example, Level 6 51 Jacobson St: 26 WELLINGTON ST E **SUITE 500** TORONTO ON M5E 1S2 This field is not used in Australia, Austria, Belgium, Brazil, Denmark, Finland, France, Germany, Ireland, Italy, Liechtenstein, Luxembourg, Malaysia, The Netherlands, Poland, Portugal, Spain, Sweden, Switzerland, and Thailand.
City	String	The city or town name. Your input address should use the official city name. This will produce the best geocoding results. For Thailand, this field contains the subdistrict (tambon).
County	String	The name of one of the following depending on the country: • **Not used**—AUT, BRA, CAN, FIN, GBR, MYS, PRT, SGP. • **Department**—FRA • **District (amphoe)**—THA • **District (fylke/counties)**—NOR • **District (poviat)**—POL • **Kommun**—SWE • **Kreis**—DEU • **Local Government Authority (LGA)**—AUS • **Province**—BEL, CHE, DNK, ESP, IRL, ITA, LIE, LUX, NLD • **Region**—NZL
FirmName	String	Company or name or place name. For example, PITNEY BOWES. **PITNEY BOWES** 4360 DUKES RD KALGOORLIE WA 6430
LastLine	String	The last line of the address. For example, KALGOORLIE WA 6430: 4360 DUKES RD **KALGOORLIE WA 6430**
Locality	String	The name of one of the following depending on the country: • **Not used**—AUS, AUT, BEL, CHE, DEU, DNK, FIN, FRA, IRL, LIE, LUX, MYS, NLD, NOR, POL, SGP, SWE, THA

columnName	Format	Description
		• **Dissemination Area and Enumeration Area (DA and EA)**—CAN • **Locality**—BRA, GBR, ITA, PRT • **Suburb**—NZL
PostalCode	String	The postal code in the appropriate format for the country.
StateProvince	String	The name of one of the following depending on the country: • **Not used**—BEL, CHE, DNK, IRL, LIE, LUX, NLD, NOR, SGP • **Bundesland**—DEU • **Province**—CAN • **Province (changwat)**—THA • **Province (voivodship)**—POL • **Region**—AUT, ESP, FRA, GBR, NZL, PRT • **Region (län)**—FIN • **Region (lan)**—SWE • **State**—AUS, BRA • **State (negeri)**—MYS
Country	String	The two- or three-character ISO country code. This field is optional. If you omit the country, GeocodeAddressWorld returns the best available candidates based on the other input provided For a list of ISO codes, see **Country ISO Codes and Module Support**.

Address Aliases

Some countries have alternative administrative names. For example, there may be an official name for a city or town, but there may also be common but unofficial alternative name for the same city or town. If alias information is available in the source data, Geocode Address World includes this alias in the database. This enables Geocode Address World to geocode successfully when alternative names are used in input addresses.

Language Aliases on page 233 are also supported.

Language Aliases

Some countries have more than one official or prominent language. For example, the same town may be commonly known by both German and Italian names. If language alias information is available in the source data, Geocode Address World uses this in the database. This enables Geocode Address World to geocode successfully when alternative language names are used in input addresses.

Aliases can exist for all administrative levels, from StateProvince state/province to Locality locality. See **Administrative Divisions and Postal Codes** on page 234 for a description of administrative levels associated with the geographic data.

Address Aliases on page 233 are also supported for commonly used, alternative administrative areas.

State or Province Abbreviations

In some countries, the state or province is an important part of the address and often this address component is abbreviated. For selected countries, these state/province abbreviations are recognized by Geocode Address World. For example, in the United States each state has a two-letter abbreviation (such as CA for California). Similarly, Netherlands, state abbreviations (such as GLD for Gelderland) are recognized.

Geocode Address World accepts state/province abbreviations for the following countries:

Table 45: Country State/Province Abbreviation Support

Country Name	State Or Province Division	Example
Australia (AUS)	StateProvince (State)	NSW (abbreviation for New South Wales
Canada (CAN)	StateProvince (Province)	AB (abbreviation for Alberta)
Italy (ITA)	County (Province)	MO (abbreviation for Modena)
Mexico (MEX)	StateProvince (State)	JA (abbreviation for Jalisco)
Netherlands (NLD	County (State)	FR (abbreviation for Friesland)
United States (USA)	StateProvince (State)	CA (abbreviation for California)

Geocode Address World evaluates these state or province abbreviations to better identify close matches. See **Geographic Geocoding with State/Province Abbreviation** on page 215 for an example that illustrates this feature.

Administrative Divisions and Postal Codes

Typical input addresses consist of street address, administrative division, and postal code information. Geocode Address World uses the administrative divisions and postal codes for geographic or postal geocoding.

- StateProvince (state or province)
- County (county, region, or district)
- City (town or city)
- Locality (locality suburb, or village)
- postal code

Specific administrative divisions vary by country. For example, Locality may contain locality, suburb, or barrio, depending on the country. StateProvince may contain state, province, region, or some other name depending on the country. See **State or Province Abbreviations** on page 233 for more information on how state/province abbreviations are interpreted by Geocode Address World.

Not all administrative divisions are used in addressing conventions for all countries. For example, in the USA, County (county) is not typically used in addresses. But for some countries, County is an important part of the address.

If your input data includes postal codes, Geocode Address Worldcan use this for postal geocoding, assuming that the source data includes postal data for the specific country.

Input Recommendations

You can optimize Geocode Address World results if you prepare and understand your input records. Follow these guidelines :

- Ensure that your input addresses are as complete and accurate as possible. If there are errors in your input addresses, Geocode Address World may still be able to geocode those addresses, but there may be more than one possible match or you may get non-close matches. If you can verify and correct any incomplete or inaccurate input addresses, you can get better results.
- Include postcodes in your input addresses if you have them. This is not required, but it allows Geocode Address World to perform postal geocoding. This may give you more accurate results for some addresses, depending on the country and on the completeness and accuracy of other address components.
- Include the country name or official three-character or two-character country ISO code in your input addresses. This is not required, but it may help Geocode Address World distinguish between similar addresses and city names that may occur in different countries.
- Format your input addresses consistently. Geocode Address World can handle input addresses in a wide variety of input formats, or can handle unformatted (single line) input. But you can get more accurate and faster results if your input addresses are consistently formatted and conform to country-specific address conventions. Even if your input address are single line (unformatted), you may get better results and performance if the address components are ordered consistently. Use the AddressLine1 input area for unformatted addresses. See **Single Line Input** on page 235

Single Line Input

Address input can be formatted into separate input fields or input can as single line input. Use AddressLine1 to enter single line input.

Single Line Geographic Geocoding

In this example, unformatted (single line) input is used. Geocode Address World analyzes single line input to identify the geographic address components (Graz in this example), and then geocodes to a geographic centroid. The MainAddress (street information) is not used.

Sackstraße 10 Graz

Geocode Address World returns a geographic close match candidate based on an City match. Even though the country was not specified, Geocode Address World identifies the close match in Austria (AUT).

StateProvince: Steirmark
County: Graz (Stadt)
City: Graz
Country: AUT
Result Code: G3
X: 15.44172
Y: 47.06792

If your input addresses are accurate, unformatted input can produce a match rate comparable to that of formatted input. However, geocoding unformatted addresses typically has slower performance than geocoding formatted addresses.

Single Line Postal Geocoding with Country Specified

In this example, single line input is used and a postcode is provided. The country Austria (AUT) is also specified. The street address is also input, but this is ignored by Austria.

Alpenstraße 117 5020 AUT

Austria returns a postal centroid close match candidate (Z1 result code). Because the country (AUT) is specified in the input, the country must be matched and a single close match for that postal code in Austria is returned. Non-close matches with the 5020 postal code from other countries are also returned.

StateProvince: Salzburg
Country: AUT
Postcode: 5020
Result Code: Z1
X: 13.04685
Y: 47.80262

Options

Geocoding Options

The following table lists the options that control how a location's coordinates are determined.

Table 46: Geocoding Options

optionName	Description
CoordinateSystem	A coordinate system is a reference system for the unique location of a point in space. Cartesian (planar) and Geodetic (geographical) coordinates are examples of reference systems based on Euclidean geometry. Spectrum™ Technology Platform supports systems recognized by the European Petroleum Survey Group (EPSG). One the following: **EPSG:4283** Also known as the GDA94 coordinate system. **EPSG:4326** Also known as the WGS84 coordinate system. Default.

Matching Options

Table 47: Matching Options

optionName	Description
KeepMultimatch	Specifies whether to return results when the address matches to multiple candidates in the database. If this option is not selected, an address that results in multiple candidates will fail to geocode. If you select this option, specify the maximum number of candidates to return using the MaxCandidates option (see below). **Y** Yes, return candidates when multiple candidates are found. Default. **N** No, do not return candidates. Addresses that result in multiple candidates will fail to geocode.
MaxCandidates	If you specify KeepMultimatch=Y, this option specifies the maximum number of results to return. The default is 1.
CloseMatchesOnly	Specifies whether to return only those geocoded results that are close match candidates. For example, if there are 10 candidates and two of them are close candidates, and you enable this option, only the two close matching candidates would be returned instead of all 10. **Y** Yes, return only close matches. **N** No, do not return only close matches. Default.

Data Options

The Data tab allows you to specify which databases to use in geocoding. Databases contain the address and geocode data necessary to determine the geocode for a given address. The data is based on address and geocoding data from postal authorities and suppliers of geographical data.

Table 48: Data Options

optionName	Description
DatabaseSearchOrder	The name of one or more database resources to use in the search process. Use the database name specified in the Management Console's Database Resources tool. For more information, see the *Spectrum™ Technology Platform Administration Guide*. You can specify multiple database resources. If you specify more than one database, list them in order of preference. The order of the database has an effect when there are close match candidates from different databases. The close matches that are returned come from the database that is first in the search list. Close matches from lower ranked databases are demoted to non-close matches.

Output

GeocodeAddressWorld returns the latitude/longitude, city, county, and result indicators. Result indicators describe how well the geocoder matched the input to a known location and assigned a latitude/longitude; they also describe the overall status of a match attempt. The output returned is in the DataTable class. For information on the DataTable class, see the "API Fundamentals" section.

Address Output

Table 49: Address Output

columnName	Description
City	Municipality name.
CityRank	CityRank is a numeric value ranging from 1 (high) to 10 (low) based on total and relative population, importance, and other criteria.
Country	The three-letter ISO 3166-1 Alpha 3 country code. The two-letter code can also be used. See **Country Geographic Data Coverage** on page 217 for a list of countries and data sources for geographic geocoding. See **Country Postal Data Coverage** on page 225 for a list of postal geocoding countries and data sources.
County	This field contains an area that is smaller than a state/province but larger than a city. The specific area varies by country: • AUS—Local Government Authority (LGA) • AUT—Province • BEL—Province • BHS—Not used • BRA—Not used • CAN—Not used • CHE—Province • DEU—Kreis • DNK—Province • FIN—Province (kommune) • FRA—Department • GBR—County

columnName	Description
	• ITA—Province • LIE—Province • LUX—Province • MYS—District (daerah) • NLD—Province • NZL—Not used • POL—District (poviat) • PRT—Not used • SGP—District • SWE—Region (kommun) • THA—District (amphoe)
PostalCode	The postcode for the address. The format of the postcode varies by country.
StateProvince	The meaning of StateProvince varies by country: • AUS—State • AUT—Region • BEL—Not used • BRA—State • CAN—Province • CHE—State • DEU—Bundesland • DNK—Not used • ESP—Region • FIN—Region (län) • FRA—Region • GBR—Region • IRL—Not used • ITA—Region • LIE—State • LUX—Not used • MYS—State (negeri) • NLD—Not used • NOR—Not used • NZL—Region • POL—Province (voivodship) • PRT—Region • SGP—Not used • SWE—Region (lan) • THA—Province (changwat)

Geocode Output

Table 50: Geocode Output

columnName	Description
CoordinateSystem	The coordinate system used to determine the latitude and longitude coordinates. A coordinate system specifies a map projection, coordinate units, etc. An example is EPSG:4326. EPSG stands for European Petroleum Survey Group.
Latitude	Seven-digit number in degrees and calculated to four decimal places (in the format specified).
Longitude	Seven-digit number in degrees and calculated to four decimal places (in the format specified).

Result Codes

Result codes contain information about the success or failure of the geocoding attempt, as well as information about the accuracy of the geocode.

Table 51: Result Code Output

columnName	Description
Geocoder.MatchCode	Indicates how closely the input address matches the candidate address.
IsCloseMatch	Indicates whether or not the address is considered a close match. An address is considered close based on the "Close match criteria" options on the Matching tab. **Y** Yes, the address is a close match. **N** No, the address is not a close match.
MultiMatchCount	For street address geocoding, the number of matching address positions found for the specified address. For intersection geocoding, the number of matching street intersection positions found for the specified addresses.
Status	Reports the success or failure of the match attempt **null** Success **F** Failure
Status.Code	If the geocoder could not process the address, this field will show the reason. • Internal System Error • No Geocode Found • Insufficient Input Data • Multiple Matches Found • Exception occurred • Unable to initialize Geocoder

columnName	Description
	• No Match Found
Status.Description	If the geocoder could not process the address, this field will show a description of the failure. **Problem + explanation** — Returned when Status.Code = Internal System Error. **Geocoding Failed** — Returned when Status.Code = No Geocode Found. **No location returned** — Returned when Status.Code = No Geocode Found. **No Candidates Returned** — The geocoder could not identify any candidate matches for the address. **Multiple Candidates Returned and Keep Multiple Matches not selected** — The address resulted in multiple candidates. In order for the candidate address to be returned, you must specify `KeepMultimatch=Y`.
LocationPrecision	A code describing the precision of the geocode. One of the following: **0** — No coordinate information is available for this candidate address. **1** — Interpolated street address. **2** — Street segment midpoint. **3** — Postal code 1 centroid. **4** — Partial postal code 2 centroid. **5** — Postal code 2 centroid. **6** — Intersection. **7** — Point of interest. This is a placeholder value. Spectrum databases do not have POI data, so it is not possible to get this return. **8** — State/province centroid. **9** — County centroid. **10** — City centroid. **11** — Locality centroid. **12 - 15 (LocationPrecision codes)** — For most countries, LocationPrecision codes 12 through 15 are reserved for unspecified custom items. **13** — Additional point precision for unspecified custom item. **14** — Additional point precision for unspecified custom item. **15** — Additional point precision for unspecified custom item. **16** — The result is an address point.

columnName	Description	
	17	The result was generated by using address point data to modify the candidates segment data.
	18	The result is an address point that was projected using the centerline offset feature. You must have both a point and a street range database to use the centerline offset feature, and thereby return LocationPrecision 18.
StreetDataType	The default search order rank of the database used to geocode the address. A value of "1" indicates that the database is first in the default search order, "2" indicates that the database is second in the default search order, and so on. The default database search order is specified in the Management Console with the Database Resources tool.	

Geographic Candidate Ranking

Identical geographic area names can be found in many countries. When this occurs, Geocode Address World uses a ranking system to determine which of potential candidates is the most likely close match.

The specific details of this weighted ranking depend somewhat on the data source (TomTom, GeoNames, or Pitney Bowes source), but the following criteria are weighted to determine the most likely close match candidate.

- country capital
- administrative area (state/province, regional, county) capital
- population range

Country capital status outweighs any other geographic ranking criteria. For example, San Juan entered as City returns San Juan, Puerto Rico (PRI) as the close match because it is the capital of PRI. Other San Juan cities in the world (including Spain, Costa Rica, Dominican Republic and Philippines) can be returned as non-close matches regardless of their population. To return matches, you must check the Keep multiple matches check box in Matching Options of the Management Console, and specify the number of matches you want to return.

Similarly, Geocode Address World returns Roma, ITA as a close match since that is the capital of Italy, but Roma in Romania, Honduras, and Panama are returned as non-close matches.

State/province administrative capitals are highly weighted even if their populations are not very large. For example, Springfield returns a close geographic match to Springfield, Illinois USA because this is the state capital of Illinois. Springfield, Massachusetts has a somewhat larger population, but this is outweighed by the state capital status of Springfield Illinois. Other less populous Springfield communities in the USA and other countries are also returned as close matches, but are listed below the Springfield Illinois candidate. It is possible for a large city to rate as an equal close match along with a smaller, identically named state/province capital. However, the state/province capital will not be demoted, even if it has a relatively small population.

Similarly, if your input is Albany in City with no country specified, Geocode Address World returns Albany, NY, USA as the close match candidate. This is because Albany is the capital of New York State, and therefore gets a high ranking as administrative area capital. The population is also a contributing ranking factor. If you specify the city of Albany with a different country, such as New Zealand, then the country is used and Albany, NZL is returned as the close match candidate.

If a candidate includes a city, a CityRank value is also returned, if available. CityRank is a numeric value ranging from 1 (high) to 10 (low) that indicates the relative ranking of the city. This ranking is based on relative population, administrative status, and other criteria. If multiple geographic candidates are returned, they are listed in city rank order.

Match Codes

Matches in the G category indicate that the candidate is located at the geographic centroid with the following possible accuracy levels. Not all levels of accuracy are possible for all countries.

- **G0**—Country centroid. This is not returned for GeocodeAddressWorld.
- **G1**—State or province centroid. For Japan, this indicates a prefecture (ken) match.
- **G2**—County centroid. For Japan, this indicates a city (shi) match.
- **G3**—City centroid. For Japan, this indicates a municipality subdivisio (oaza) match. For Australia, Local Government Authority (LGA) information can be returned from the Street Range Address Database only (not the G-NAF database).
- **G4**—Locality centroid. For Japan, this indicates a city district (chome) match.

Matches in the Z category indicate that no street match was made for one of the following reasons:

- You specified to match to postal code centroids. The resulting point is located at the postal code centroid with four possible accuracy levels.
- There is no close match and you specified to fall back to postal code centroid

The Z category contains the following accuracy levels:

- **Z0**—Postal Code match, no coordinates available (rare occurrence).
- **Z1**—Postal Code centroid match.
- **Z3**—Full postal code centroid match. For Canada, this is an FSALDU centroid.
- **Z6**—Postal Code centroid match for point ZIP.

GeocodeUSAddress

GeocodeUSAddress takes an address and returns latitude/longitude coordinates. GeocodeUSAddress also standardizes and validates addresses using data from the U.S. Postal Service.

GeocodeUSAddress can also geocode intersections. Instead of entering a mailing address, you can enter and intersection such as "Pearl St. and 28th" and obtain the coordinates of the intersection.

GeocodeUSAddress is part of the Enterprise Geocoding Module. For more information on the Enterprise Geocoding Module, including a listing of other components included with it, see **Enterprise Geocoding Module** on page 204.

Input

GeocodeUSAddress takes an address as input. To obtain the best performance with GeocodeUSAddress and the most possible matches, your input address should be as complete as possible and free of misspellings and incomplete information. Input addresses should be as close to USPS standards as possible for the highest match rate. For information on USPS standards, see the USPS website **http://www.usps.com** .

Input addresses should contain a street address line and a lastline, or a single line with both address and lastline elements. This helps GeocodeUSAddress accurately identify an area in which to search for a match candidate, based on the city, state, and ZIP Code. However, matching can still be performed in cases where only a city is input in the lastline rather than a full lastline containing the city, state and ZIP Code elements. For more information, see **City-only Lastline Matching** on page 245.

GeocodeUSAddress also accepts a street address line with individual city, state, and ZIP Code lines instead of a last line. You should only use this type of input if you are confident that the input address is free of misspellings and incomplete information.

If you are using GeocodeUSAddress for address standardization, input addresses must have at least a street name, and either a city and state or a ZIP Code to obtain a match. If you are using GeocodeUSAddress to obtain geocoding information, input addresses only need to contain a ZIP + 4 Code to receive geocoding information.

The following table provides information on the format and layout of GeocodeUSAddress input.

Table 52: GeocodeUSAddress Input Data

columnName	Format	Description
AddressLine1	String	The first address line or a street intersection. To specify a street intersection, use and, &, at, or @. For example, PEARL & 28th. GeocodeUSAddress does not match intersections when processing in CASS mode. You may enter an address range instead of an individual address number. For example, 10-12 FRONT ST. For additional information, see **Address Range Matching** on page 245.
AddressLine2	String	The second address line or a street intersection. To specify a street intersection, use and, &, at, or @. For example, PEARL & 28th.L GeocodeUSAddress does not match intersections when processing in CASS mode.
AddressLine3	String	Third address line.
AddressLine4	String	Fourth address line.
AddressLine5	String	Fifth address line.
AddressLine6	String	Sixth address line.
City	String	The name of the municipality, such as a city or town. **Note:** If there is any data in the input fields AddressLine3, AddressLine4, AddressLine5, or AddressLine6, GeocodeUSAddress will ignore data in the City input field.
FirmName	String	The name of a business. GeocodeUSAddress attempts to match the input firm name to the recognized firm names in the USPS data for a higher quality match. If the firm name is not in the USPS data, GeocodeUSAddress ignores the firm name when matching and returns the firm name with the output.
LastLine	String	The last line of an address containing the city, state, and ZIP Code.
Latitude	String	Input latitude for matching using the Predictive Lastline feature. The required input format is an integer value; for example, 40018301. Decimal is not a valid input format. For more information, see **Predictive Lastline** on page 246.
Longitude	String	Input longitude for matching using the Predictive Lastline feature. The required input format is an integer value; for example: -105240976. Decimal is not a valid input format. For more information, see **Predictive Lastline** on page 246.
PostalCode	String	The 5-digit ZIP Code or the 9-digit ZIP + 4 code. **Note:** If there is any data in the input fields AddressLine3, AddressLine4, AddressLine5, or AddressLine6,

columnName	Format	Description
		GeocodeUSAddress will ignore data in the PostalCode input field.
StateProvince	String	The name or abbreviation of the state. **Note:** If there is any data in the input fields AddressLine3, AddressLine4, AddressLine5, or AddressLine6, GeocodeUSAddress will ignore data in the StateProvince input field.

How GeocodeUSAddress Processes Addresses

GeocodeUSAddress processes addresses in the following order:

1. Parses the address elements.

   GeocodeUSAddress parses input address data into single elements. Parsing occurs on data in the order in which you load the data. Even if a valid address is missing an element, GeocodeUSAddress can find a match. Some elements, such as predirectionals, may not be critical elements of some addresses. By comparing an address as input against all known addresses in a search area, GeocodeUSAddress can usually determine if any of these elements are missing or incorrect.

2. Finds possible matches within the search area.

   GeocodeUSAddress uses the last line elements of an address to determine a search area. You can specify if you want the search area based on a finance area or on an area defined by the city, state, and ZIP Code. (A Finance Area is a collection of ZIP Codes within a contiguous geographic region.) If the city and state are not in the ZIP Code, GeocodeUSAddress performs separate searches for the ZIP Code and city.

   After GeocodeUSAddress has determined the search area, it tries to match the elements from the street address line to the records in the standardized data files and does the following:

   - Checks input address ranges for missing or misplaced hyphens, and alpha-numeric ranges for proper sequence.
   - Searches for any misspellings and standard abbreviations. For example, the GeocodeUSAddress can recognize Mane for Main and KC for Kansas City.
   - Searches for any alias matches to the USPS and Spatial data (TIGER and TomTom). For example, GeocodeUSAddress recognizes that in Boulder, CO Highway 36 is know as 28th Street.
   - Searches for any USPS recognized firm names for additional match verification.
   - Searches for street intersection matches. Matching to an intersection is extremely useful when you are using address matching to obtain a geocode.
   - Searches for addresses lines that contain a house number and unit number as the same element. For example, GeocodeUSAddress recognizes the input 4750-200 Walnut Street and performs recombination to output 4750 WALNUT ST STE 200.

   Note: The USPS does not consider intersections valid addresses for postal delivery. Therefore, the GeocodeUSAddress does not match intersections when processing in CASS mode.

3. Scores each possible match against the parsed input.

   GeocodeUSAddress compares each element in the input address to the corresponding element in the match candidates, and assigns a confidence level. GeocodeUSAddress weighs the confidence level for all of the elements within a match candidate, and assigns a final score to the sum.

   Note: GeocodeUSAddress uses a penalty scoring system. If an element does not exactly match an element in the match candidate, the GeocodeUSAddress adds a penalty to the score of the match candidate. Therefore, scores with lower numbers are better matches.

4. Determines the match.

GeocodeUSAddress prioritizes each match candidate based on the assigned confidence score and returns as a match the candidate that has the lowest score.

The match mode you choose determines the range that GeocodeUSAddress allows for a match. GeocodeUSAddress only returns a match if the score of the target address falls within the range designated by the selected match mode.

In some cases, more than one match candidate may have the lowest score. In this instance, GeocodeUSAddress cannot determine on its own which record is correct, and returns a status indicating multiple matches.

Note: If you have enabled Delivery Point Validation (DPV) processing, GeocodeUSAddress automatically attempts to resolve multiple matches using DPV.

Along with a standardized address, GeocodeUSAddress also returns the following:

- Geocode—Longitude and latitude for the address
- Match code—Information about the match of the input address to the reference data
- Location code—Precision level of a geocode
- Parity—The side of the street on which the match resides.

GeocodeUSAddress does not return parity when processing in relaxed mode. For more information on GeocodeUSAddress output, see **Output** on page 259.

City-only Lastline Matching

City-only lastline matching permits address matching with only a city in the input lastline. The city should be provided using either the `AddressLine1` (using single-line address input), `AddressLine2`, `LastLine` or `City` input fields.

With city-only lastline input, all of the states are searched in which the input city exists. Therefore, there is the possibility of an increase in multimatches (return of E023 or E030 Match Codes) when matching with city-only input instead of city+state input.

Restrictions:

- City-only lastlne input matching is not supported in CASS mode.
- City-only lastline is not supported when matching to User Dictionaries.
- When matching using city-only lastline, the PreferZipCodeOverCity setting is ignored
- It is strongly recommended to not use city-only lastline matching in Relaxed match mode to avoid the return of false-positive matches.

Address Range Matching

Some business locations are identified by address ranges. For example, a shopping plaza could be addressed as 10-12 Front St. This is how business mail is typically addressed to such a business location. These address ranges can be geocoded to the interpolated mid-point of the range.

Address ranges are different from hyphenated (dashed) addresses that occur in some metropolitan areas. For example, a hyphenated address in Queens County (New York City) could be 243-20 147 Ave. This represents a single residence (rather than an address range) and is geocoded as a single address. If a hyphenated address returns as an exact match, GeocodeUSAddress does not attempt to obtain an address range match.

Address range matching is not available in Exact or CASS modes, since an address range is not an actual, mailable USPS® address. The following fields are not returned by address range geocoding:

- ZIP + 4® (in multiple segment cases)
- Delivery point
- Check digit
- Carrier route
- Record type
- Multi-unit
- Default flag

Address range matching works within the following guidelines:

- There must be two numbers separated by a hyphen.
- The first number must be lower than the second number.
- Both numbers must be of the same parity (odd or even) unless the address range itself has mixed odd and even addresses.
- Numbers can be on the same street segment or can be on two different segments. The segments do not have to be contiguous.
- If both numbers are on the same street segment, the geocoded point is interpolated to the approximate mid-point of the range.
- If the numbers are on two different segments, the geocoded point is based on the last valid house number of the first segment. The ZIP Code and FIPS Code are based on the first segment.
- In all cases, odd/even parity is evaluated to place the point on the correct side of the street.

Predictive Lastline

Predictive lastline allows you to match an address when only an input street address and latitude/longitude coordinates are provided, rather than the traditional street address with lastline input. For example, an input of 4750 Walnut with latitude/longitude coordinates located in Boulder, will return full address information.

Enabling Predictive Lastline

The following tables describes how to enable and set up the Predictive Lastline feature.

Directions
Set `PredictiveLastLine = Y`
Set `Latitude = integer value`
Set `Longitude = integer value`
Set `AddressLine1 = input street address`

Additional feature information

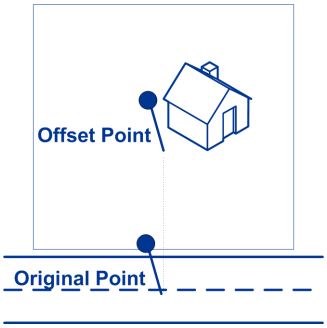
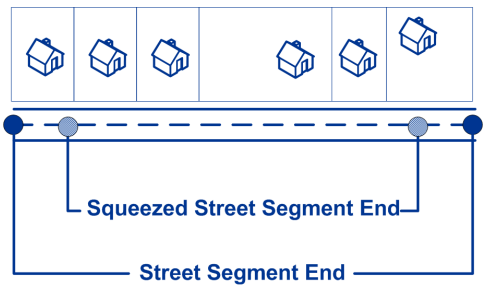
- Predictive Lastline uses a search radius of 150 feet.
- Predictive Lastline is disabled by default.
- When Predictive Lastline is not enabled, if a singleline address is provided, any input latitude and longitude values are ignored and the input address is geocoded.
- If the input latitude/longitude coordinates fall near the borders of multiple cities, the Enterprise Geocoding Module processes all cities and returns the results of the best match. If the results are determined as equal, then a multi-match is returned.
- Predictive lastline does not require a license for reverse geocoding.
- This feature will work with any type of data set.

Options

Geocoding Options

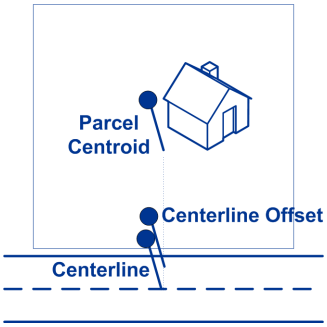
Table 53: GeocodeUSAddress Geocoding Options

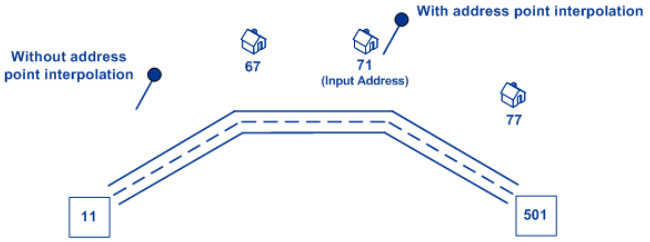
optionName	Description
Dataset	The name of the database resource that contains the data to use in the search process. Use the database name specified in the Management Console's Database Resources tool. For more

optionName	Description
	information, see the *Spectrum™ Technology Platform Administration Guide*.
Offset	Specifies the offset distance from the street segments, in feet. The range is 0 to 5280. The default value is 50. The offset distance is used in street-level geocoding to prevent the geocode from being in the middle of a street. It compensates for the fact that street-level geocoding returns a latitude and longitude point in the center of the street where the address is located. Since the building represented by an address is not on the street itself, you do not want the geocode for an address to be a point on the street. Instead, you want the geocode to represent the location of the building which sits next to the street. For example, an offset of 50 feet means that the geocode will represent a point 50 feet back from the center of the street. The distance is calculated perpendicular to the portion of the street segment for the address. Offset is also used to prevent addresses across the street from each other from being given the same point. The following diagram shows an offset point in relation to the original point.  Street coordinates are accurate to 10,000ths of a degree and interpolated points are accurate to the millionths of a degree.
Squeeze	Specifies the distance, in feet, to move the street segment end points toward the middle of the segment. Squeeze is used in street-level matching. Use the squeeze setting to prevent address points from residing in an intersection or too close to the end of a street. The range is 0 to 2147483647. The default value is 50. The following diagram compares the end points of a street to squeezed end points.  Squeezing the street segment endpoints affects street-level matching by reducing the length of a street segment, thereby

optionName	Description
	reducing the spacing between address points along the segment. For example, if the length of a street segment is 1,000 feet and there are 10 addresses along the segent, street-level matching would result in each address being spaced 100 feet apart (1,000 ÷ 10). If you were to set a squeeze value of 100 feet, moving each street segment endpoint 100 feet torward the center of the street segment, the length of the street segment would be reduced to 800 feet (reduced by 100 feet on each end). Street-level matching would then result in addresses beging spaced 80 feet apart (800 ÷ 10).
LatLonFormat	Specifies the format of the latitude/longitude returned by GeocodeUSAddress. **Decimal** The latitude/longitude is returned in decimal format (default). For example: 90.000000-180.000000 **Integer** The latitude/longitude is returned in integer format. For example: 90000000-180000000
CentroidPreference	Determines the type of centroids returned by GeocodeUSAddress. A centroid is the center of an area. The centroid coordinates are the average of the sets of coordinates that describe the area. **NoCentroids** Do not return centroids. If an address-level geocode cannot be determined, do not attempt to determine a centroid. **AllCentroids** Return ZIP Code centroids only. If you select this option, address-level geocodes will not be returned. **AddressUnavailable** Return a ZIP Code centroid if an address-level geocode cannot be determined (default).
FallbackToStreet	Specifies whether to attempt to return a street centroid when an address-level geocode cannot be determined. To determine a street centroid, GeocodeUSAddress searches the input ZIP Code or city for the closest match. If GeocodeUSAddress is able to locate the street, it returns a geocode along the matched street segment. For example, if the input address is 5000 Walnut Street, Boulder 80301, and there is no 5000 Walnut Street, GeocodeUSAddress searches for the closest match to that address within the ZIP Code 80301. If there were no input ZIP Code, GeocodeUSAddress would search for the closest match to the input address within Boulder. If the input address is Walnut Street, Boulder 80301, since there is no house number, GeocodeUSAddress searches for the street within the input ZIP Code. Street centroid geocodes are indicated by value in the LocationCode output field that begins with "C". For more information, see **Street Centroid Location Codes** on page 334. **Note:** This option is not available if you set MatchMode to CASS.

optionName	Description
	Y Yes, attempt to determine the street centroid when an address-level geocode cannot be determined. **N** No, do not attempt to determine the street centroid when an address-level geocode cannot be determined.
FallbackToGeographic	Specifies whether to attempt to return a city, county, or state centroid when an address-level geocode cannot be determined. The geocoder returns the most precise geographic centroid that it can based on the input. For example, if the input contains a valid city and state, a city centroid would be returned. **Note:** There are approximately 300 major cities that can be geocoded to a city centroid level even if a valid state is not provided in the input. Geographic centroid geocodes are indicated by value in the LocationCode output field that begins with "G". For more information, see **Geographic Centroid Location Codes** on page 339. **Note:** This option is not available if you set MatchMode to CASS. **Y** Yes, attempt to determine the geographic centroid when an address-level geocode cannot be determined. **N** No, do not attempt to determine the geographic centroid when an address-level geocode cannot be determined.
Datum	Determines the North American Datum to use when geocoding datum on the input value. Datum is the mathematical model of the Earth used to calculate the coordinates on any map, chart, or survey system. **NAD27** This datum does not include the Alaskan Islands or Hawaii. Latitudes and longitudes that are surveyed in the NAD27 system are valid only in reference to NAD27 and are not valid for maps outside the U.S. **NAD83** This datum is earth-centered and defined with satellite and terrestrial data. NAD83 is compatible with the World Geodetic System 1984 (WGS84), which is the terrestrial reference frame associated with the NAVSTAR Global Positioning System (GPS) used extensively for navigation and surveying. This is the default setting.
CenterlineOffset	The offset distance, in feet, used to calculate the street centerline coordinates. The default is 0. If you specify a value other than 0, GeocodeUSAddress calculates the street centerline coordinates by offsetting the centerline point by the distance you specify in the direction of the parcel centroid.

optionName	Description
	 In an interpolated match, the centerline offset cannot be greater than the distance from the centerline to the interpolated address point. If you specify a centerline offset distance that is greater than this distance, the offset will be limited to the distance to the interpolated point. In effect, the centerline coordinates would be the same as the coordinates for the interpolated point.
RetrieveElevation	Specifies whether GeocodeUSAddress returns the elevation of the address. Elevation is the distance above or below sea level of a given location. The elevation is returned in the **Elevation** output field, which is part of the Latitude/Longitude output group. **Note:** This option requires that you have licensed and installed the Centrus Premium Points database. Elevation data is not available for all addresses. See the coverage map included with the points database.
AddressPointInterpolation	Specifies whether to perform address point interpolation when an exact match for the address cannot be found in the geocoding database. Address point interpolation is a patented process that results in a more accurate interpolated point. It improves upon regular street segment interpolation by using point data in the interpolation process, as opposed to using street segments alone. **Note:** Address point interpolation is only available when using a point-level geocoding database. It is not available when using point addresses in an auxiliary file. **Y** Yes, perform address point interpolation. **N** No, do not perform address point interpolation. The following illustration shows how address point interpolation works. In the example, the input house number is 71. The geocoding database contains address points for 67 and 77. The street segment has a range of 11 to 501. With address point interpolation, GeocodeUSAddress performs the interpolation for the input house number 71 using the points of 67 and 77. Without address point interpolation, GeocodeUSAddress performs the interpolation with the street segment end points of 11 and 501, resulting in a far less accurate result.

optionName	Description
	

Matching Options

Table 54: GeocodeUSAddress Matching Options

optionName	Description
AddressPreference	Determines which address to use when more than one address is present in the address block. **PreferBottom** — Uses the second line entered (default). You must select this value if you specify MatchMode=CASS. **PreferPOBox** — Uses the P.O Box. **PreferStreetAddress** — Uses the street address.
FirmNameSearch	Specifies whether GeocodeUSAddress should use firm name matching logic to enhance address matching. Firm matching logic matches a business name in the input to recognized business names. The input firm name does not need to be spelled correctly to obtain a match. GeocodeUSAddress uses a soundex algorithm to match the firm name. A suite or unit number is not required to make the match. **Note:** This type of match is not available when processing in CASS mode. One of the following: **Never** — Do not use firm matching (default). Note that GeocodeUSAddress may correct the firm name even if you specify **Never** if it can find a match using the address line data. **OnAddressLineFail** — Use firm matching only if a match cannot be determined using address matching. **Always** — Always attempt to match using firm name matching. If firm name matching fails, attempt to match using address matching.
FirstLetterSearch	Specifies whether to look for the correct first letter of a street name if the first letter is missing or incorrect. If enabled, GeocodeUSAddress searches through the alphabet looking for the correct first letter to complete the street address.

optionName	Description
	Note: This option is not available if the match mode is set to exact. **Y** Perform first letter search. **N** Do not perform first letter search (default). This example includes an incorrect first letter: Input: 4750 nalnut boulder co 80301 Output: 4750 Walnut St Boulder CO 80301-2532 This example excludes a first letter: Input: 4750 alnut boulder co 80301 Output: 4750 Walnut St Boulder CO 80301-2532 This example includes an extra first letter: Input: 4750 wwalnut boulder co 80301 Output: 4750 Walnut St Boulder CO 80301-2532
BuildingSearch	Specifies whether GeocodeUSAddress attempts to obtain a street address when the input address contains a building name with no suite or unit number. When this option is disabled, GeocodeUSAddress is able to match to building names only if there is a unit number in the input. For example, if the building search option were disabled and you entered this input: 5001 Chrysler Bldg New York, NY 10174 GeocodeUSAddress would successfully return the street address: 405 Lexington Ave RM 5001 New York, NY 10174-5002 With this option enabled, GeocodeUSAddress is also able to obtain a street address when only a building name with no unit number is provided. For example, if you enable this option and provide this address: Chrysler Bldg New York, NY 10174 You will get the street address: 405 Lexington Ave New York, NY 10174-00 **Note:** This type of match is not available when processing in CASS mode. **Y** Use firm name matching logic (default). **N** Do not use firm name matching logic.
RetrieveAPN	Specifies whether GeocodeUSAddress should determine the address's APN (assessor's parcel number). The APN is an ID

optionName	Description
	number assigned to a property by the local property tax authority. The APN is returned in the **APN** output field, which is part of the Census output group. **Note:** This option requires that you have licensed and installed the Cenrus Enhanced Points or Centrus Premium Points database. APN data is not available for all addresses. See the coverage map included with the points database.
PredictiveLastLine	Specifies whether GeocodeUSAddress should match using the street address and input latitiude/longitude coordinates, rather than the traditional street address with lastline input. **Y** Enable Predictive Lastline processing. **N** Disable Predictive Lastline processing (default). For more information, see **Predictive Lastline** on page 246.
PerformDPV	Specifies whether GeocodeUSAddress should process addresses using Delivery Point Validation (DPV). DPV is a United States Postal Service (USPS) technology that validates the accuracy of address information down to the physical delivery point. You must have licensed the optional DPV processing option to use this feature. You must also install the DPV database. To use DPV, enable this processing option and specify D in OutputRecordType. **Y** Perform DPV. **N** Do not perform DPV (default). If you use DPV, GeocodeUSAddress automatically resolves multiple matches. False-positive addresses, also known as seed records, are addresses the USPS monitors to ensure users are not attempting to create a mailing list from the DPV data. If GeocodeUSAddress matches an address in your input data to a false-positive address, you receive a message indicating you have encountered a false-positive address. Processing continues to the end of your job, but DPV processing is not available for this job and subsequent jobs until you have reported the false-positive address encounter to technical support and have received a new security key.
PerformLACSLink	Specifies whether GeocodeUSAddress should process addresses using LACS^{Link}. **Y** Perform LACS^{Link} **N** Do not perform LACS^{Link} (default). If you use LACS^{Link}, be sure to choose to specify output record types P and Q so that the fields USLACS, USLACS.ReturnCode, and LACSADDRESS are included in the output. For more information, see **Locatable Address Conversion System (LACS)** on page 211.

optionName	Description
PreferZipCodeOverCity	Specifies whether to prefer candidates that match the input ZIP over candidates that match to input city. **Note:** This option is not available when processing in CASS mode. **Y** Perfer candidates that match the input ZIP Code. **N** Prefer candidates that match the input city (default). For example, consider this input address: 301 BRYANT ST SAN FRANCISCO CA 94301 Without this option enabled, the best match would be the one that matches the input city name: 301 BRYANT ST SAN FRANCISCO CA 94107-4167 With this option enabled, the best match would be the one that matches the input ZIP Code: 301 BRYANT ST PALO ALTO CA 94301-1408
KeepMultimatch	Select this option to return the list of possible matches when Geocode US Address finds more than one possible match for the input address and cannot identify a single best match. **Y** Return the addresses that are possible matches for the input address (default). **N** Do not return the ambiguous matches.
KeepCandidates	Select this option to return candidate addresses whenever the match attempt produces candidates. If you enable this option, the geocoder will return candidates both when the input address matches to a single address and when the input address matches multiple addresses. This option differs from `KeepMultimatch` in that the `KeepMultimatch` option does not return candidates if the input address matches to a single address. **Y** Return candidates for all match attempts. **N** Do not return candidates for all matches (default).
CloseMatchesOnly	If you specify `KeepCandidates=Y` you can choose to return just those candidates that are considered to be a close match. The criteria used to determine whether a candidate is a close match are those you specify in the MatchMode option. **Note:** A close match does not necessarily indicate a high-quality match. You should always check the values in the output fields MatchCode and LocationCode to determine the quality of the match. **Y** Return close match candidates only (default).

optionName	Description	
	N	Return all candidates.
MatchMode	Determines the leniency used to find a match. One of the following:	
	Custom	Allows you to select the specific criteria to use when matching the input address to an address in the postal database.
	Exact	Requires a very tight match. This is a restrictive mode that generates the fewest number of match candidates to search, which decreases the time to obtain a match. When using this mode, ensure that your input address list is very clean; free of misspellings and incomplete addresses.
	Close	Requires a moderately confident match. Generates a moderate number of match candidates.
	Relax	Default. This is the loosest match mode and generates the most match candidates, which increases the processing time and results in more multiple matches. Use this mode if your address list may contain misspellings and incomplete addresses. This mode does not respect the street parity for an address match.
	Interactive	Available in single-line address matching only. This mode is designed to better handle the specific matching challenges presented by interactive matching. Interactive mode permits for more flexible matching patterns and may, in some cases, return additional possible matches than relaxed match mode. This mode recognizes and parses two unit numbers on the same address line, for example a building and unit number. This mode does not respect the range parity when making an address match. Capabilities and restrictions: • Interactive match mode allows users to break the cardinal rule: If the user enters 123 S Main and there is only 123 N Main, a match is made and a match code is returned that reflects the modified directional. • Interactive match mode handles cases where users transpose pre-directionals with post-directionals without penalty. • Interactive match mode ignores the 'Prefer ZIP Code over city' setting. When the city and ZIP Code don't match correctly, the best geocoding result will be returned based on an analysis of all the input address elements. • When operating in interactive mode, in cases where a point address or interpolated street address result cannot be determined, ZIP-9 or ZIP-7 centroid(s) may be returned.

optionName	Description
	CASS Imposes additional rules to ensure compliance with the USPS regulations for CASS. The purpose of this mode is to create a list of mailable addresses. This mode generates a large number of match candidates. This mode deviates from the other modes in its processing. This mode does not perform intersection, building name, or spatial alias (TIGER and TomTom street name alias) matches. It also does not match to candidates from data sources that do not have USPS equivalent records. This mode recognizes and parses two unit numbers on the same address line, for example a building and unit number.
MustMatchInput	Specifies whether candidates must match all non-blank input fields. For example, if an input address contains a city and postal code, then candidates for this address must match the city and postal code. **Y** Yes, candidates must match all input. **N** No, candidates do not have to match all input. Default.
MustMatchHouseNumber	Specifies whether candidates must match the house number. If the input house number is not within a range from the street, GeocodeUSAddress selects the nearest range on the street which has the same parity (even or odd house number) as the input address number. GeocodeUSAddress returns one or more of the closest matches inside this range that preserves street parity. This requires GeocodeUSAddress to change the house number. The new house number is equal to one of the range's endpoints, possibly plus or minus one to preserve street parity. **Note:** Even when this option is disabled and an inexact match on the house number is found, GeocodeUSAddress still returns an error code. When this option is disabled and no exact matching house number is found, a match code of either E029 (no matching range, single street segment found), or E030 (no matching range, multiple street segment) is returned. GeocodeUSAddress does not change the house number on the output address. In order to access the inexact address number candidates, you must specify `KeepMultimatch=Y`. If there are inexact house number candidates returned, the corresponding match codes begin with the letter 'H' indicating that the house number was not matched. Additionally, even when one or more exact candidates are found, inexact matches to the house number are still on the list of possible candidates, and these can be differentiated from the others by their Hxx match codes. For more information on match codes, see **Geocoding Match Codes** on page 341. One of the following: **Y** Yes, candidates must match the house number.

optionName	Description
	N No, candidates do not have to match the house number.
MustMatchStreet	Specifies whether candidates must match the street name. **Y** Yes, candidates must match the street name. **N** No, candidates do not have to match the street name.
MustMatchCity	Specifies whether candidates must match the city. If you do not require exact matches on city, the geocoder searches on the street address matched to the particular postal code, and considers other cities that do not match the name, but do match the postal code. **Y** Yes, candidates must match the city. **N** No, candidates do not have to match the city.
MustMatchStateProvince	Specifies whether candidates must match the state. **Y** Yes, candidates must match the state. **N** No, candidates do not have to match the state.
MustMatchPostalCode	Specifies whether candidates must match the postal code. If you do not require exact match on postal codes, the geocoder searches a wider area for a match. While this results in slower performance, the match rate is higher because the request does not need to match exactly when it compares match candidates. **Y** Yes, candidates must match the postal code. **N** No, candidates do not have to match the postal code.

Difference Between Match Criteria for U.S. and Non-U.S. Geocoding

The "must match criteria" used in the custom match mode of Geocode US Address work differently than the "close match criteria" in non-U.S. geocoders. For Geocode US Address, the custom match criteria specify which address elements must match the reference database in order for the match to be returned as a candidate. All candidates returned by Geocode US Address will match the elements you specify as long as those elements are available in the reference database. However, in non-U.S. geocoders, the "close match" criteria are used to determine which candidates are close matches and which are non-close matches. Non U.S. geocoders can return both close candidates and non-close candidates, depending on whether you enable the `CloseMatchesOnly` option. In summary, the "must match" criteria used by Geocode US Address automatically limit the candidates returned, whereas the "close match criteria" used by non-U.S. geocoders do not limit the candidates returned.

Output Format

The following table lists the GeocodeUSAddress options that control the format of the output.

Table 55: GeocodeUSAddress Output Format Options

optionName	Description
OutputCasing	Specifies the casing of the output data. One of the following:

optionName	Description
	M The output in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345
OutputFormattedOnFail	Specifies whether GeocodeUSAddress normalizes an addresses that fail to match, and addresses that are unchanged. Normalization formats an address to the USPS guidelines without validating the address. **Y** Perform standardization (default). **N** Do not perform standardization.
OutputPostalCodeSeparator	Specifies whether GeocodeUSAddress should include the dash in full postal code output. **Y** Include the dash (default). **N** Do not include the dash.
OutputVerbose	Specifies whether GeocodeUSAddress provides an additional description field as output. These fields provide the text equivalent to a field represented by a code. For example, LocationCode returns a code that indicates the accuracy (quality) of the assigned geocode. LocationCode.Description provides the description for the code returned. **Y** Include verbose fields. **N** Do not include verbose fields (default).

Output Data

The following table shows the GeocodeUSAddress options that control which data GeocodeUSAddress returns in the output.

Table 56: GeocodeUSAddress Output Data Options

optionName	Description
OutputRecordType	Specifies optional data to include in the output. Note that GeocodeUSAddress always returns the default data listed in **Default Output** on page 267. The data you select here is returned with the default output data. • **B**—Block Address • **C**—Census • **D**—DPV (The DPV field group is disabled unless you select **Perform Delivery Point Validation (DPV)** on the **Configuration Options** tab.) • **Z**—Geoconfidence • **L**—Latitude/Longitude

optionName	Description
	• **E**—Parsed Elements • **P**—Postal Data • **Q**—Qualifiers • **R**—Range • **S**—Segment For a description of the fields in each output group, see **Output** on page 259. If you do not want all of the fields in a particular category returned, do not select the check box, and list only those fields you want returned in OutputFields.
OutputFields	Specifies the individual output fields you want returned. List fields with a pipe (|) between each field. You can use this option instead of the OutputRecordType option to limit the output to those fields that are important to your data needs. By default, these are the address fields returned: AddressLine1|LastLine|Longitude|Latitude|MatchCode|LocationCode For a list of all the fields included in each data field, see **Output** on page 259.

Output

GeocodeUSAddress always returns a default set of output fields that contain the latitude/longitude, standardized address, and result indicators. For information on these fields, see **Default Output** on page 267. You can also choose to include the following optional categories of output data:

Auxiliary

Auxiliary data output fields contain information on the an auxiliary file match. For more information on using an auxiliary file, see **Auxiliary File Overview** on page 323. GeocodeUSAddress only returns values when matching against an auxiliary file. To include segment data fields in the output, set **OutputRecordType** = X.

Table 57: Auxiliary Data Output Fields

columnName	Max. Field Length (bytes)	Description
AuxiliaryData	301	The user data field in an auxiliary file match. **Note:** GeocodeUSAddress does not process this information. It simply includes the user data contained in the auxiliary file.
MCDCode	6	The Minor Civil Division (MCD) code. A Minor Civil Division is a subdivision of a county, such as a township. There are Minor Civil Divisions in 28 states, the District of Columbia, Puerto Rico, and Island Areas. Minor Civil Divisions are defined by U.S. Census Bureau.
MCDName	41	The Minor Civil Division (MCD) name. A Minor Civil Division is a subdivision of a county, such as a township. There are

columnName	Max. Field Length (bytes)	Description
		Minor Civil Divisions in 28 states, the District of Columbia, Puerto Rico, and Island Areas. Minor Civil Divisions are defined by U.S. Census Bureau.

Block Address

Block data output fields contain extraneous information from the input address that GeocodeUSAddress could not process. To include block data in the output, set **OutputRecordType** = B.

If there are any empty lines in the input fields AddressLine1 through AddressLine6, GeocodeUSAddress moves the output lines to the first empty BlockLine output field, eliminating the blank lines. For example:

Input Field	Input Data	Output Field	Output Data
		AddressLine1	4750 Walnut St Ste 200
		LastLine	Boulder, CO 80301-2532
AddressLine1	Pitney Bowes	BlockLine1	Pitney Bowes
AddressLine2	4750 Walnut	BlockLine2	
AddressLine3		BlockLine3	
Data.AddressLine3			
AddressLine4	Ste 200	BlockLine4	Dept ABC **Note:** Moved up one line from the input AddressLine5.
AddressLine5	Dept ABC	BlockLine5	
AddressLine6	80301	BlockLine6	
Data.AddressLine6			

Table 58: Block Data Output Fields

columnName	Max. Field Length (bytes)	Description
BlockLine1	104	Returns input address information GeocodeUSAddress could not process.
BlockLine2	104	Returns input address information GeocodeUSAddress could not process.

columnName	Max. Field Length (bytes)	Description
BlockLine3	104	Returns input address information GeocodeUSAddress could not process.
BlockLine4	104	Returns input address information GeocodeUSAddress could not process.
BlockLine5	104	Returns input address information GeocodeUSAddress could not process.
BlockLine6	104	Returns input address information GeocodeUSAddress could not process.

Census

Census output fields contain U.S. Census information about the address. To include census data in the output, set **OutputRecordType** = C.

Table 59: Census Data Output Fields

columnName	Max. Field Length (bytes)	Description
APN	46	The assessor's parcel number of the property. The assessor's parcel number is an ID number assigned to a property by the local property tax authority.
BlockSuffix	2	The block suffix for the Census block in which the address is located. A block suffix is a single character assigned to subsections of U.S. Census blocks that are split by a higher-level boundary, such as a municipal boundary. A block suffix is either "A" or "B". For information about U.S. Census block suffixes, see the *Geographic Areas Reference Manual*, available at the U.S. Census Bureau website: **www.census.gov/geo/www/garm.html** Block suffixes are only available if you are using Centrus Enhanced data.
CBSACode	6	The code for the Core Based Statistical Area (CBSA) in which the address is located. A CBSA is a collective term that refers to both metropolitan and micropolitan areas. A metropolitan area has a population of more than 50,000, and a micropolitan area has a population between 10,000 and 49,999. For more information, see *Metropolitan and Micropolitan Statistical Areas* section of the U.S. Census Bureau website: **www.census.gov/population/www/metroareas/metroarea.html**

columnName	Max. Field Length (bytes)	Description
CBSADivisionCode	6	The code of the Core Based Statistical Area (CBSA) division in which the address is located. A CBSA division is a metropolitan statistical area with a population of at least 2.5 million that has been subdivided to form smaller groupings of counties referred to as "metropolitan divisions." For more information, see *Metropolitan and Micropolitan Statistical Areas* section of the U.S. Census Bureau website: **www.census.gov/population/www/metroareas/metroarea.html**
CBSADivisionName	128	The name of the Core Based Statistical Area (CBSA) division in which the address is located. A CBSA division is a metropolitan statistical area with a population of at least 2.5 million that has been subdivided to form smaller groupings of counties referred to as "metropolitan divisions." For more information, see *Metropolitan and Micropolitan Statistical Areas* section of the U.S. Census Bureau website: **www.census.gov/population/www/metroareas/metroarea.html** The CBSA division name is only returned if you set the option OutputVerbose=Y.
CBSAMetro	2	Indicates whether the core based statistical area (CBSA) in which the address is located is a metropolitan area or a micropolitan area. One of the following: **Y** Yes, the address is located in a metropolitan statistical area. Metropolitan areas have a population greater than 50,000. **N** No, the address is not located in a metropolitan area. It is located in a micropolitan area. Micropolitan areas have a population between 10,000 and 49,999. **null** There is no data available to determine whether the address is in a metropolitan or micropolitan area. A CBSA is a collective term that refers to both metropolitan and micropolitan areas. A metropolitan area has a population of more than 50,000, and a micropolitan area has a population between 10,000 and 49,999. For more information, see *Metropolitan and Micropolitan Statistical Areas* section of the U.S. Census Bureau website: **www.census.gov/population/www/metroareas/metroarea.html**
CBSAName	128	The name of the core based statistical area (CBSA) in which the address is located. The CBSA division name is only returned if you set the option OutputVerbose=Y. A CBSA is a collective term that refers to both metropolitan and micropolitan areas. A metropolitan area has a population of more than 50,000, and a micropolitan area has a population

columnName	Max. Field Length (bytes)	Description
		between 10,000 and 49,999. For more information, see *Metropolitan and Micropolitan Statistical Areas* section of the U.S. Census Bureau website: **www.census.gov/population/www/metroareas/metroarea.html**
CensusBlockID	16	The 15-digit identification number of the census block in which the address is located. Census blocks are the smallest geographic area for which the Bureau of the Census collects and tabulates decennial census data. Census blocks are formed by streets, roads, railroads, streams and other bodies of water, other visible physical and cultural features, and the legal boundaries shown on Census Bureau maps. For more information about U.S. Census blocks, see the *Geographic Areas Reference Manual*, available at the U.S. Census Bureau website: **www.census.gov/geo/www/garm.html** The Census block ID is in the format: `ssccctttttgbbb` Where: **ss** The two-digit state FIPS code. **ccc** The three-digit county FIPS code. **ttttt** The six-digit Census tract FIPS code. **g** The single-digit block group FIPS code. **bbb** The block FIPS code. **Note:** GeocodeUSAddress does not return a period for the Centrus tract FIPS code. This may deviate from the industry standard.
CensusTract	7	The six-digit ID of the Census tract in which the address is located. Census tracts are small, relatively permanent geographic entities within counties (or the statistical equivalents of counties). Generally, census tracts have between 2,500 and 8,000 residents and boundaries that follow visible features. For more information about U.S. Census tracts, see the *Geographic Areas Reference Manual*, available at the U.S. Census Bureau website: **www.census.gov/geo/www/garm.html**
CSACode	4	Denotes the code for a geographic entity that consists of 2 or more adjacent CBSAs with employment interchange measures of at least 15.
CSAName	128	The name of the combined statistical area (CSA) in which the address is located. A CSA is a combination of two or more adjacent Core Based Statistical Areas (CBSAs) with a high employment interchange

columnName	Max. Field Length (bytes)	Description
		measure. The employment interchange measure is the sum of the percentage of employed residents of the smaller entity who work in the larger entity and the percentage of the employment in the smaller entity that is accounted for by workers who reside in the larger entity. Pairs of CBSAs with employment interchange measures of at least 25% combine automatically. Pairs of CBSAs with employment interchange measures of at least 15%, but less than 25%, may combine if local opinion in both areas favors combination. The CSA name is only returned if you set the option OutputVerbose=Y.
USCountyName	128	The name of the county or parish in which the address is located. The county/parish name is only returned if you set the option OutputVerbose=Y.
USFIPSCountyNumber	4	The three-digit FIPS county code of the county in which the address is located.
USFIPSStateCode	3	The two-digit FIPS state code of the state in which the address is located.
USFIPSStateCountyCode	6	The five-digit FIPS code for the state and county in which the address is located.

Centerline Projection

Centerline projection output fields contain information specific to a centerline match. To include centerline projection fields in the output, set **OutputRecordType** = N.

Table 60: Centerline Projection Output Fields

columnName	Max. Field Length (bytes)	Description
CenterlineBearing	6	The compass direction, in decimal degrees, from the point data match to the street centerline match. The compass direction is measured clockwise from 0 degrees north. For example, if the centerline match is directly north of the point match, the centerline bearing would be 0.
CenterlineBlockLeft	16	The Census FIPS Code that indicates the address is on the left side of the street.
CenterlineBlockRight	16	The Census FIPS Code that indicates the address is on the right side of the street.
CenterlineBlockSuffixLeft	2	The block suffix of the block on the left side of the street.

columnName	Max. Field Length (bytes)	Description
		A block suffix is a single character assigned to subsections of U.S. Census blocks that are split by a higher-level boundary, such as a municipal boundary. A block suffix is either "A" or "B". For information about U.S. Census block suffixes, see the *Geographic Areas Reference Manual*, available at the U.S. Census Bureau website: **www.census.gov/geo/www/garm.html** Block suffixes are only available if you are using Centrus Enhanced data.
CenterlineBlockSuffixRight	2	The block suffix of the block on the right side of the street. A block suffix is a single character assigned to subsections of U.S. Census blocks that are split by a higher-level boundary, such as a municipal boundary. A block suffix is either "A" or "B". For information about U.S. Census block suffixes, see the *Geographic Areas Reference Manual*, available at the U.S. Census Bureau website: **www.census.gov/geo/www/garm.html** Block suffixes are only available if you are using Centrus Enhanced data.
CenterlineDataCode	3	Indicates the data used to obtain a centerline match for the address. One of the following: **0** USPS data in either the Centrus Enhanced, Centrus TomTom, or Centrus NAVTEQ database. **1** TIGER data in the Centrus Enhanced database. **2** TomTom data in the Centrus TomTom database. **6** NAVTEQ data in the Centrus NAVTEQ database. **7** TomTom point-level data in the Centrus TomTom Points database. **8** Point-level data from the Centrus Points database. **9** Auxiliary file data. **10** User Dictionary. **11** NAVTEQ point-level data. **12** Master Location data.

columnName	Max. Field Length (bytes)	Description
		For more information about these databases, see **Enterprise Geocoding Databases** on page 205
CenterlineDirection	2	Indicates the order of numbers on a segment for a centerline match. **F** Forward **R** Reversed **B** Both **U** Undetermined
CenterlineDistance	8	Distance, in feet, from the point-level match to the centerline match.
CenterlineHouseNumberHigh	12	The highest address number in the range of addresses on the street segment. For example, if the address range for the street segment is 1000 to 2000, the CenterlineHouseNumberHigh would be 2000.
CenterlineHouseNumberLow	12	The lowest address number in the range of addresses on the street segment. For example, if the address range for the street segment is 1000 to 2000, the CenterlineHouseNumberLow would be 1000.
CenterlineIsAlias	4	Three characters indicating that GeocodeUSAddress located a centerline match by an index alias. The first is an `N` for normal street match or `A` for alias match (including buildings, aliases, firms, etc.). The next two characters are: **01** Basic index (normal address match) **02** USPS street name alias index **03** USPS building index **04** USPS firm name index **05** Statewide intersection alias match (when using the Usw.gsi or Use.gsi file) **06** Spatial data street name alias (when using, the Us_pw.gsi, Us_pe.gsi, Us_psw.gsi, or Us_pse.gsi file is required) **07** Alternate index (when using Zip9.gsu, Zip9e.gsu, and Zip9w.gsu) **08** LACS^{Link} **09** Auxiliary file match **10** Centrus Alias index (when using usca.gsi)

columnName	Max. Field Length (bytes)	Description
CenterlineLatitude	11	A seven-digit number in degrees and calculated to four decimal places for a centerline match. This field is only returned if AlwaysFindCandidates=Y
CenterlineLeadingDirectional	3	The street directional that precedes the street name for a centerline match. For example, the N in 138 N Main Street.
CenterlineLongitude	12	7-digit number in degrees and calculated to 4 decimal places (in the format specified) for a centerline match. This field is only returned if AlwaysFindCandidates=Y
CenterlineParity	2	Indicates which side of the street has odd numbers for a centerline match. **L** The left side of the street has odd numbers. **R** The right side of the street has odd numbers. **B** Both sides of the street have odd numbers. **U** Undetermined.
CenterlineRoadClass	3	The type of road for a centerline match: **1** Major **2** Minor
CenterlineSegmentCode	11	The unique 10-digit street segment ID assigned by the street network data provider.
CenterlineStreetName	41	The name of the street.
CenterlineStreetSuffix	5	The street type of the matched centerline location. For example, AVE in "Washington AVE".
CenterlineTrailingDirectional	3	The street directional that follows the street name. For example, the N in 456 Washington AVE N.

Default Output

GeocodeUSAddress always returns fields that contain the latitude/longitude, standardized address, and result indicators. Result indicators describe how well GeocodeUSAddress matched the input address to a known address and assigned a location. They also describes the overall status of a match attempt.

Table 61: Default Output Fields

columnName	Max. Field Length (bytes)	Description
AdditionalInputData	61	This field is populated with information from the mail stop output field.

columnName	Max. Field Length (bytes)	Description
		Note: GeocodeUSAddress does not process this information. It simply includes the information as entered in the input data.
AddressLine1	104	The first line of the address. For example: **1 Global View** Troy, NY 12180-8371
AddressLine2	104	The second line of the address. For example: 4200 Parliament Pl **STE 600** Lanham, MD 20706-1882
City	29	The municipality name.

Confidence	4	Indicates the confidence in the output provided, from 0 to 100. The higher the score, the higher the probability that the match is correct. If the match is exact, the confidence score is 100. For all other matches, GeocodeUSAddress calculates the confidence score by subtracting values from 100 as follows:
		
		• If GeocodeUSAddress changed the state to obtain a match:
		  • Added the state -3.75
		  • No state -7.5
		• If GeocodeUSAddress changed the city to obtain a match:
		  • Added city -2.5
		  • No city -5.0
		• If GeocodeUSAddress change the house number to obtain a match:
		  • Added house number -3.75
		  • No house number -7.5
		• If GeocodeUSAddress changed the street name to obtain a match:
		  • Added street name -3.75
		  • No street name -7.5
		• If GeocodeUSAddress changed the trailing directional to obtain a match:
		  • Added trailing directional -1.25
		  • No trailing directional -2.5
		• If GeocodeUSAddress changed the leading directional to obtain a match:
		  • Added leading directional -1.25
		  • No leading directional -2.5

columnName	Max. Field Length (bytes)	Description

		• If GeocodeUSAddress changed the street suffix to obtain a match:
		  • Added street suffix -1.25
		  • No street suffix -2.5
		• If GeocodeUSAddress changed the postal code to obtain a match: -11.25
		If you have enabled the option to return centroids, the confidence value indicates the type of centroid returned:
		• 60 for a street centroid
		• 50 for a postal code centroid
		• 35 for a city centroid
		• 30 for a county centroid
		• 25 for a state centroid

Country	25	The name of the country. Since GeocodeUSAddress only works for U.S. locations, this field will always contain **United States of America**.
FirmName	41	The name of the business if the address is a business address.
LastLine	61	The complete last address line (city, state, and postal code).
Latitude	11	Seven-digit number in degrees and calculated to four decimal places (in the format specified).
LocationCode	5	A value indicating the accuracy (quality) of the assigned geocode. For more information, see **Address Location Codes** on page 328.
Longitude	12	Seven-digit number in degrees and calculated to four decimal places (in the format specified).
MatchCode	5	Indicates the portions of the address that matched to the directory file. For more information, see **Geocoding Match Codes** on page 341.
PostalCode	10	Nine-digit ZIP Code with or without a hyphen.
PostalCode.AddOn	5	Four-digit ZIP Code extension
PostalCode.Base	6	Five-digit ZIP Code.
ProcessedBy	4	The underlying software that processed the request. EnterpriseGeocoding for GeocodeUSAddress.
StateProvince	3	Two-character state abbreviation.
Status	2	Reports the success or failure of the match attempt

columnName	Max. Field Length (bytes)	Description
		null Success **F** Failure
Status.Code	23	If GeocodeUSAddress could not process the address, this field will show the reason. • Internal System Error • No Geocode Found • Insufficient Input Data
Status.Description	128	If GeocodeUSAddress could not process the address, this field will show a description of the failure. **Problem + explanation** Returned when Status.Code = Internal System Error. **Geocoding Failed** Returned when Status.code = No Geocode Found. **No location returned** Returned when Status.code = No Geocode Found.
StreetDataType	20	The data set GeocodeUSAddress attempted to match against. **USPS** USPS **TIGER** TIGER **TOMTOM** TomTom Streets data set **SANBORN POINT DATA** Sanborn Points data set **NAVTEQ** NAVTEQ Streets data set **TOMTOM POINT DATA** TomTom Points data set **AUXILIARY** Auxiliary file **CENTRUS POINT DATA** Centrus Points data set **USER DICTIONARY** User Dictionary **NAVTEQ POINT DATA** NAVTEQ Points data set **MASTER LOCATION** Master Location Data
StreetSide	2	Indicates the side of the street the range occupies. One of the following: **L** The range occupies the left side of the street. **R** The range occupies the right side of the street. **B** The range occupies both sides of the street. **U** Undetermined.
USUrbanName	31	Urbanization name. Used for addresses in Puerto Rico.

DPV

DPV data output fields contain information about a match made using DPV data. GeocodeUSAddress only returns values when matching against DPV data. To include DPV data in the output, set **OutputRecordType** = D.

Table 62: DPV Data Output Fields

columnName	Max. Field Length (bytes)	Description
CMRA	2	Indicates whether the address is for a Commercial Mail Receiving Agent (CMRA). A CMRA is a private company that rents out mailboxes. A customer of a commercial mail receiving agency can receive mail and other deliveries at the street address of the CMRA rather than the customer's own street address. Depending on the agreement between the customer and the CMRA, the CMRA can forward the mail to the customer or hold it for pickup. **Y** Yes, the address is a CMRA. **N** No, the address is not a CMRA. **null** DPV data is not available. DPV data is required to determine if an address is a CMRA.
DPV	2	Indicates whether the address is confirmed to be a deliverable address by USPS Delivery Point Validation (DPV). **N** Nothing confirmed **Y** Everything confirmed (ZIP+4, primary, and secondary) **S** ZIP+4 and primary (house number) confirmed **D** ZIP+4 and primary (house number) confirmed and a default match **U** Non-matched input address to USPS ZIP+4 data, or DPV data not loaded

DPVFootnote	3 per footnote	Contains detailed information about the address. The DPV footnote codes are combined together consecutively.
		DPV footnotes include the following:
		• FOOTNOTE1 provides information about matched DPV records.
		  • **AA**—ZIP+4 matched record
		  • **A1**—Failure to match a ZIP+4 record
		  • **null**—Address not presented to hash table or DPV data not loaded
		• FOOTNOTE2 provides information about matched DPV records.
		  • **BB**—All DPV categories matched

columnName	Max. Field Length (bytes)	Description
		• **CC**—Matched primary/house number, where the secondary/unit number did not match (present but invalid) • **M1**—Missing primary/house number • **M3**—Invalid primary/house number • **N1**—Matched primary/house number, with a missing highrise secondary number • **P1**—Missing PS, RR, or HC Box number • **P3**—Invalid PS, RR or HC Box number • **F1**—All military addresses • **G1**—All general delivery addresses • **U1**—All unique ZIP Code addresses • **null**—Address not presented to hash table or DPV data not loaded • FOOTNOTE3 provides information about matched DPV records. • **R1**—Matched CMRA, without a present secondary/unit number • **RR**—Matched CMRA • **null**—Address not presented to hash table or DPV data not loaded **Note:** A unique ZIP Code is a ZIP Code assigned to a company, agency, or entity with sufficient mail volume to receive its own ZIP Code.

Geoconfidence

Geoconfidence data output fields contain information about the type of geoconfidence polygon returned. To include geoconfidence fields in the output, set OutputRecordType = Z.

columnName Response Element	Max. Field Length (bytes)	Description
GeoConfidenceCode	13	The value returned in this field indicates which geoconfidence surface type has been returned. Possible values are: **INTERSECTION** A geocode point for the intersection of two streets. **ADDRESS** An array of street segment points representing the street segment where the address is located. **POINT** If the geocoder was able to match the address using point data, the point

columnName Response Element	Max. Field Length (bytes)	Description
		geometry where the address is located.
		POSTAL1 A geocode point for the ZIP centroid.
		POSTAL2 An array of points for all street segments in the ZIP + 2 in which the address is located.
		POSTAL3 An array of points for street segments in the ZIP + 4 in which the address is located.
		ERROR An error has occurred.
StreetSegmentPoints	1024	An array of latitude/longitude values that represent the street segment points. **Note:** This field contains values only if the `GeoConfidenceCode` field returns a value of `ADDRESS`, `POSTAL2`, or `POSTAL3`.
GeoConfidenceCentroidLatitude	11	The latitude of the centroid of the geoconfidence polygon.
GeoConfidenceCentroidLongitude	12	The longitude of the centroid of the geoconfidence polygon.

Latitude/Longitude

The latitude/longitude output fields contain the geographic coordinates of the address. To include latitude/longitude output fields in the output, set **OutputRecordType** = L.

Table 63: Latitude/Longitude Output Fields

columnName	Max. Field Length (bytes)	Description
Elevation	11	The location's elevation in feet above or below sea level.
Latitude	11	The latitude of the address. The latitude is a seven-digit number in degrees, calculated to six decimal places.
Longitude	12	The longitude of the address. The longitude is a seven-digit number in degrees, calculated to six decimal places.

Parsed Elements

The parsed elements output fields contain standard address information as individual units, such as street suffixes (for example AVE, ST, or RD) and leading directionals (for example N and SE). To include parsed elements in the output, set **OutputRecordType** = E.

Table 64: Parsed Elements Output Fields

columnName	Max. Field Length (bytes)	Description
ApartmentLabel	5	The type of unit, such as apartment, suite, or lot.
ApartmentLabel2	5	The type of unit, such as apartment, suite, or lot, for addresses that contain two units, such as: 123 E Main St APT 3, 4th **Floor** .
ApartmentNumber	12	Apartment number. For example: 123 E Main St APT **3**
ApartmentNumber2	12	Secondary apartment number. For example: 123 E Main St APT 3, **4th** Floor
CrossStreetLeadingDirectional[1]	3	Leading directional, for example: 123 **E** Main St Apt 3
CrossStreetName	41	Name of cross street.
CrossStreetSuffix[1]	5	Street suffix, for example: 123 E Main **St** Apt 3
CrossStreetTrailingDirectional[1]	3	Trailing directional, for example: 123 Pennsylvania Ave **NW**
HouseNumber	12	Building number for the address.
HouseNumber2	12	If an address consists of a range of house numbers, this field contains the second house number. The HouseNumber field contains the first number. For example, given this address: 5-7 Maple Ave. The HouseNumber field would contain "5" and the HouseNumber2 field would contain "7".
LeadingDirectional	3	Leading directional, for example: 123 **E** Main St Apt 3
PrivateMailbox	9	Private mailbox. Not returned for multiline input.
PrivateMailbox.Designator	5	Private mailbox description. Not returned for multiline input.
RRHC	2	Rural Route/Highway Contract portion of the address.
StreetName	41	Street name.
StreetSuffix	5	The street type of the matched location. For example, AVE for Avenue.
TrailingDirectional	3	Street directional that follows the street name. For example, the N in 456 Washington N.

[1] GeocodeUSAddress only returns Cross street outputs if you entered an intersection as an address. For example, entering Pearl and 28th, Boulder, CO returns cross street information. Entering 2800 Pearl, Boulder, CO does NOT return cross street information.

Postal Data

Postal data output fields contain detailed postal information for the address, such as the preferred city name and the US carrier route. To include postal data fields in the output, set **OutputRecordType** = P.

Table 65: Postal Data Output Fields

columnName	Max. Field Length (bytes)	Description
CityPreferredName	29	The USPS® preferred city name for the ZIP Code of the address.
CityShortName	29	The USPS®-approved abbreviation for the city, if there is one. The USPS® provides abbreviations for city names that are 14 characters long or longer. City abbreviations are 13 characters or less and can be used when there is limited space on the mailing label. If there is no short city name for the city, then the full city name is returned.
CityStateRecordName	29	USPS® city state city name.
DeliveryPointCode	3	Two-digit delivery point bar code.
GovernmentBuilding	2	Indicates if a building is used by the city, state, or federal government. **A** City government building **B** Federal government building **C** State government building **D** Firm only **E** City government building and firm only **F** Federal government building and firm only **G** State government building and firm only The values A, B, C, E, F, and G are valid for Alternate records only. The value D is valid for both base and alternate records.
PostalBarCode	7	Six-digit combination of ZIP+4 Code and the delivery point bar code.
PostalCodeClass	2	ZIP Classification code. **null** Standard ZIP Code **M** Military ZIP Code **P** ZIP Code has P.O. boxes only **U** Unique ZIP Code (ZIP Code assigned to a single organization)
PostalCodeUnique	2	Indicates if the ZIP Code is a unique ZIP Code assigned to an individual company or agency. **Y** Unique ZIP name **null** No unique ZIP name

columnName	Max. Field Length (bytes)	Description
PostalFacility	2	USPS City State Name Facility code. **A** Airport Mail Facility (AMF) **B** Branch **C** Community Post Office (CPO) **D** Area Distribution Center (ADC) **E** Sectional Center Facility (SCF) **F** Delivery Distribution Center (DDC) **G** General Mail Facility (GMF) **K** Bulk Mail Center (BMC) **M** Money Order Unit **N** Non-Postal Community Name, Former Postal Facility, or Place Name **P** Post Office **S** Station **U** Urbanization
USBCCheckDigit	2	Check-digit for delivery point bar code.
USCarrierRouteCode	5	Carrier Route code.
USCarrierRouteSort	2	Indicates if the USPS uses a carrier route sort, and what type of sort the USPS allows. **A** Automation cart allowed, optional cart merging allowed **B** Automation cart allowed, no optional cart merging allowed **C** No automation cart allowed, optional cart merging allowed **D** No automation cart allowed, no optional cart merging allowed
USCityDelivery	2	Indicates if has city-delivery carrier routes. **Y** Has city-delivery carrier routes **N** Does not have city-delivery carrier routes.
USLACS	2	Indicates if LACS^{Link} match occurred. **Y** Matched LACS^{Link} record **N** LACS^{Link} match not found **F** False-positive LACS^{Link} record **S** Secondary information (unit number) removed to make a LACS^{Link} match

columnName	Max. Field Length (bytes)	Description
		null Records not processed through LACS^{Link} For more information, see **Locatable Address Conversion System (LACS)** on page 211.
USLACS.ReturnCode	3	Indicates LACS^{Link} results. **A** Matched LACS^{Link} record **00** LACS^{Link} match was not found **09** Matched to highrise default, but noLACS^{Link} conversion **14** Found LACS^{Link} match, but no LACS^{Link} conversion **92** Secondary information (unit number) was removed to make a LACS^{Link} match **null** Records not processed through LACS^{Link} For more information, see **Locatable Address Conversion System (LACS)** on page 211.
USLOTCode	2	A combination of the 4-digit Line of Travel (LOT) Code and the ascending (A) or descending (D) indicator.

Qualifiers

Qualifier output fields contain qualification information on the match, such as the location code and the match code. To include postal data fields in the output, set OutputRecordType = Q.

Table 66: Qualifier Output Fields

columnName	Max. Field Length (bytes)	Description
AddressLineResolved	5	For two-line addresses, indicates which address line was used to obtain the match. One of the following: **0** The address could not be matched, or the address matched to multiple addresses. **1** AddressLine1 was used to obtain the match. **2** AddressLine2 was used to obtain the match. **3** Both address lines were used in their original order. **4** Both address lines were used but the order of the lines was switched to obtain the match. **5** The input address was a one line address.

columnName	Max. Field Length (bytes)	Description
CountryLevel	2	The category of postal data available. Always returns A in GeocodeUSAddress—Validates, corrects, and provides missing postal code, city name, state/county name, street address elements, and country name.
DatabaseVersion		USPS publish date, in the format `Month Year`.
EWSMatch	2	Indicates if GeocodeUSAddress denied a match because of the input address matched an address in the Early Warning System (EWS) data. **Y** The address matched to an address in the EWS data so the match was denied. **null** The address did not match to an address in the EWS data.
ExpirationDate	11	Date the database expires, in the format `MM/DD/YYYY`.
Geocoder.MatchCode	4	Indicates how closely the input address matches the candidate address. **Note:** The match codes returned in this field are different from the match codes described in **Geocoding Match Codes** on page 341. Instead, the match codes returned in this field are taken from a set of match codes that are compatible with all other country geocoders. For more information, see **Result Codes for International Geocoding** on page 348.
GeoStanMatchScore	13	Record matching score (for multimatches only).
Intersection	2	Indicates if GeocodeUSAddress found a cross-street match. **T** True **F** False
IsAlias	4	GeocodeUSAddress located a match by an index alias. Returns 3 characters. The first is an `N` for normal street match or `A` for alias match (including buildings, aliases, firms, etc.). The next 2 characters are: **01** Basic index (normal address match) **02** USPS street name alias index **03** USPS building index **04** USPS firm name index **05** Statewide intersection alias match **06** Spatial data street name alias **07** Alternate index **08** LACS^{Link} **09** Auxiliary file match

columnName	Max. Field Length (bytes)	Description
		10 Centrus Alias index (when using usca.gsi)
IsCloseMatch	2	Indicates whether or not the address was a unique match or if there were candidate addresses. **Y** Yes, the address is a close match. This field always contains "Y" if there is only one match. **N** No, the address is not a close match. The record is a candidate.
LACSAddress	2	Indicates if GeocodeUSAddress converted an address due to the Locatable Address Conversion System (LACS) **L** Converted **null** Not converted
LocationCode.Description	128	LocationCode converted to text. Only returned when you set the configuration options to return additional descriptions (verbose).
MatchCode.Description	128	MatchCode converted to text. Only returned when you set the configuration options to return additional descriptions (verbose).
RecordType	18	Indicates the record type: • GeneralDelivery • HighRise • FirmRecord • Normal • PostOfficeBox • RRHighwayContract • Geographic (non USPS TIGER match) • Auxiliary (match to an auxiliary file)
RecordType.Default	2	Indicates type of match that occurred for the record type HighRise or RRHighwayContract: **Y** Default match **N** Exact match **U** Not matched
StreetDataCode	3	Indicates the data used to geocode the address. One of the following: **0** USPS data in either the Centrus Enhanced, Centrus TomTom, or Centrus NAVTEQ database. **1** TIGER data in the Centrus Enhanced database. **2** TomTom data in the Centrus TomTom database. **6** NAVTEQ data in the Centrus NAVTEQ database.

columnName	Max. Field Length (bytes)	Description
		7 TomTom point-level data in the Centrus TomTom Points database. **8** Point-level data from the Centrus Points database. **9** Auxiliary file data. **10** User Dictionary. **11** Point-level data from NAVTEQ Points database. **12** Master Location Data (MLD). For more information about these databases, see **Enterprise Geocoding Databases** on page 205.
StreetDataType	20	Indicates the data initially used for the match attempt. Note that the output field StreetDataCode shows which data was actually used to obtain the match. The data indicated in StreetDataType may be different from that in StreetDataCode if a match cannot be made in the initial match attempt. For example, if a points database is loaded, GeocodeUSAddress will first attempt a match to the point data because this is the most accurate type of match. If a point-level match cannot be made, GeocodeUSAddress will attempt to match to street data. If the match is made using street data, then the SreetDataType would indicate the point-level data and the StreetDataCode would indicate the street data. For more information, see **How GeocodeUSAddress Processes Addresses** on page 244.

Range

Range output fields contain information on the street range, such as the high and low unit numbers. To include range data fields in the output, set **OutputRecordType** = R.

Table 67: Range Data Output Fields

columnName	Max. Field Length (bytes)	Description
Alternate	2	USPS code that specifies whether a record is a base or alternate record. **B** Base record. Base records can represent a range of addresses or an individual address, such as a firm record. **A** Alternate record. Alternate records are individual delivery points.
HouseNumberHigh	12	The highest house number in the range.

columnName	Max. Field Length (bytes)	Description
HouseNumberLow	12	The lowest house number in the range.
HouseNumberParity	2	Indicates if the house number range contains even or odd numbers. **E** Even **O** Odd **B** Both
PostalCodeExtensionHigh	5	The highest four-digit ZIP Code extension in the range. The ZIP Code extension is the four digits at the end of the ZIP Code. For example: 60510-**1134**.
PostalCodeExtensionLow	5	The lowest four-digit ZIP Code extension in the range. The ZIP Code extension is the four digits at the end of the ZIP Code. For example: 60510-**1134**.
UnitNumberHigh	12	The highest unit number in the range.
UnitNumberLow	12	The lowest unit number in the range.
UnitNumberParity	2	Indicates if the unit number range contains even or odd numbers. **E** Even **O** Odd **B** Both

Segment

Segment output fields contain information on the street segment identified by the data provider. To include segment data fields in the output, set **OutputRecordType** = S.

Table 68: Segment Data Output Fields

columnName	Max. Field Length (bytes)	Description
BlockLeft	16	The Census FIPS Code that indicates the address is on the left side of the street.
BlockRight	16	The Census FIPS Code that indicates the address is on the right side of the street.
BlockSuffixLeft	2	The block suffix of the block on the left side of the street. A block suffix is a single character assigned to subsections of U.S. Census blocks that are split by a higher-level boundary, such as a municipal boundary. A block suffix is either "A" or "B". For information about

columnName	Max. Field Length (bytes)	Description
		U.S. Census block suffixes, see the *Geographic Areas Reference Manual*, available at the U.S. Census Bureau website: **www.census.gov/geo/www/garm.html** Block suffixes are only available if you are using Centrus Enhanced data.
BlockSuffixRight	2	The block suffix of the block on the right side of the street. A block suffix is a single character assigned to subsections of U.S. Census blocks that are split by a higher-level boundary, such as a municipal boundary. A block suffix is either "A" or "B". For information about U.S. Census block suffixes, see the *Geographic Areas Reference Manual*, available at the U.S. Census Bureau website: **www.census.gov/geo/www/garm.html** Block suffixes are only available if you are using Centrus Enhanced data.
RoadClass	3	The type of road: **1** Major **2** Minor
SegmentCode	11	The unique 10-digit street segment ID assigned by the street network data provider.
SegmentDirection	2	Indicates the order of numbers on a segment. **F** Forward **R** Reversed **B** Both **U** Undetermined
SegmentHouseNumberHigh	12	The highest house number in the segment.
SegmentHouseNumberLow	12	The lowest house number in the segment.
SegmentLength		The length, in feet, of a block segment. *This field is deprecated*.
SegmentParity	2	Indicates which side of the street has odd numbers. **L** Left side of the street **R** Right side of the street **B** Both sides of the street **U** Undetermined

columnName	Max. Field Length (bytes)	Description
StreetSide.NAVTEQ	2	Indicates which side of the street the address is located on. The value in this field is determined by using the NAVTEQ reference nodes for the street segment. A street segment represents part of a street. Each segment has a node at each end: the reference node and the non-reference node. The reference node is the node with the lower latitude (southernmost). If the latitudes of both nodes are identical, the reference node is the end node with the lower longitude (westernmost). The street side corresponds to the street sides you would see if you were standing at the reference node and looking at the non-reference node. One of the following: **L** The address is on the left side of the street. **R** The address is on the right side of the street. **B** The address occupies both sides of the street. **U** The street side is unknown. **null** NAVTEQ data was not used, or segment output data was not selected, or the address did not match a street segment (for example, the address was geocoded to a centroid).

Reports

There is one report available with GeocodeUSAddress: the GeocodeUSAddress Summary Report. To create the report, in Enterprise Designer, drag the **GeocodeUSAddress Summary Report** icon to the canvas. You do not need to draw a connector to the report.

GeocodeUSAddress Summary Report

The GeocodeUSAddress Summary Report contains information about the job, such as the settings, number of records processed, performance statistics, and the database used, as well as detailed information about the geocoding and address matching results. This report contains the following sections.

Address Matching Summary

This section shows the number of records processed and the number of matches obtained.

- **Total Records in File**—The total number of records in the input file used by this job.
- **Records Processed**—The number of input records minus those records that were skipped.
- **Addresses Matched**—The number of addresses that were successfully standardized and validated. This count includes all the Standardized addresses listed under the Matched Address Records section plus those with match codes beginning with G (auxiliary file), T (geographic data only), and X (intersections).
- **Unmatched**—The number of records that could not be validated.

Matched Address Records

This section contains information about the addresses that were successfully matched.

- **Standardized**—The number of addresses that match to USPS-relevant records. These addresses have match codes beginning with A (alias), D (small town), Q (unique ZIP), S (street), and U (rare). Only these types of matches are counted as Standardized.
- **Auxiliary File**—The number of records that were matched to a user-defined file. These records have a MatchCode beginning with G.
- **Intersections**—The number of records that were matched to an intersection. These records have a MatchCode beginning with X.
- **Non-USPS**—The number of records that were matched to non-USPS data.

Unmatched Address Records

This section lists the number of unmatched addressees and the reasons why the addresses were not matched. For information on these codes, see **Geocoding Match Codes** on page 341.

Standardized Address Quality

This section describes the changes that GeocodeUSAddress made to addresses in order to validate them.

- **Original address unchanged**—None of the address elements were changed to obtain a match.
- **Original last line unchanged**—The last line (city, state, ZIP Code) was unchanged but other elements were changed to obtain a match.
- **Corrected predirectional**—The predirectional of a street name was changed to obtain a match. For example, E MAIN ST changed to W MAIN ST.
- **Corrected street name**—The name of the street was changed to obtain a match. For example, MAIN ST changed to MAINE ST.
- **Corrected street suffix**—The street suffix was changed to obtain a match. For example, MAIN ST changed to MAIN AVE.
- **Corrected postdirectional**—The postdirectional of a street was changed to obtain a match. For example, MAIN ST NW changed to MAIN ST SW.
- **Corrected city name**—The name of the city was corrected to obtain a match. For example, LOS ANGLES changed to LOS ANGELES.
- **Corrected state abbreviation**—The state abbreviation was corrected to obtain a match. For example, ROCHESTER NY changed to ROCHESTER MN.
- **Corrected ZIP Code**—The ZIP Code as corrected to obtain a match. For example: 1071 MAPLE LN BATAVIA IL 49423 Changed to: 1071 MAPLE LN BATAVIA IL 60510
- **Corrected ZIP + 4 add on**—The four digits that appear after the "-" in a ZIP + 4 were corrected to obtain a match. For example, 60510 changed to 60510-1135.

Geocode Matching Levels

This section describes the level of accuracy for the geocodes returned by GeocodeUSAddress.

- **Geocodes Assigned**—The number of records to which GeocodeUSAddress assigned a geocode. This is the cumulative count of the following fields:
- **Address Match**—Address geocodes indicate a geocode made directly to a street network segment (or two segments, in the case of an intersection). Addresses included in this count have a value that begins with A in the **LocationCode** output field.
- **Auxiliary Match**—The geocode was determined using the Auxiliary File. Addresses included in this count have a value that begins with AG in the **LocationCode** output field. For more information, see **Auxiliary Match Details** on page 285.
- **Point Match**—The geocode was determined using a points database, which means the geocode represents the center of a building or parcel. Addresses included in this count have a value that begins

with AP in the **LocationCode** output field. For more information, see **Point Matching Details** on page 285.

- **ZIP Centroids Match**—The address could not be matched, so the geocode is the center of the address's ZIP Code. This is the least accurate geocode for a given address. Addresses included in this count have a value that begins with Z in the **LocationCode** output field. For more information, see **ZIP Centroid Matching Details** on page 286.

Auxiliary Match Details

This section describes the level of accuracy for the geocodes returned by GeocodeUSAddress that were obtained from the Auxiliary File. For more information, see **Auxiliary File Overview** on page 323.

These fields are ordered from the most accurate type of geocode to the least.

- **Point**—The geocode represents the center of a building or parcel. Addresses included in this count have a value of AG0 in the **LocationCode** output field.
- **Centerline**—The geocode represents the location of an address along a street segment. Addresses included in this count have a value of AG1 in the **LocationCode** output field.
- **Centerline with unknown street side**—The geocode represents the location of an address along a street segment but the side of the street where the address resides could not be determined. Addresses included in this count have a value of AG2 in the **LocationCode** output field.
- **Midpoint**—The geocode represents the midpoint of the street segment where the address resides. GeocodeUSAddress could not determine where on the street segment the address is located. Addresses included in this count have a value of AG3 in the **LocationCode** output field.

Point Matching Details

This section describes the types of point-level geocodes returned by GeocodeUSAddress. Point-level geocodes represent the center of a parcel or building.

- **User Dictionary Centroid**—The geocode is returned by a User Dictionary. Addresses included in this count have a value of AP00 in the **LocationCode** output field.
- **Parcel Centroid**—The geocode represents the center of a parcel. Addresses included in this count have a value of AP02 in the **LocationCode** output field.
- **Field-collected GPS**—The geocode is determined using data collected by teams of field verification specialists who drive the roads of selected areas to verify and update the data. Addresses included in this count have a value of AP04 in the **LocationCode** output field.
- **Structure Centroid**—The geocode represents the center of an addressable building footprint. An addressable structure is typically a structure that receives mail or has telephone service. Addresses included in this count have a value of AP05 in the **LocationCode** output field.
- **Manual Frontage Midpoint**—The geocode represents the center of a the parcel's boundary with the street. These points are offset at a specific distance from the street centerline near the center of the side of the parcel that fronts the street segment. Street frontage points estimate address locations more accurately than do interpolated ranges. Addresses included in this count have a value of AP07 in the **LocationCode** output field.
- **Front Door Point**—The geocode represents the designated primary entrance to a building. If a building has multiple entrances and there is no designated primary entrance or the primary entrance cannot readily be determined, the primary entrance is chosen based on proximity to the main access street and availability of parking. Addresses included in this count have a value of AP08 in the **LocationCode** output field.
- **Driveway Offset Point**—The geocode represents a point located on the primary access road (most commonly a driveway) at a perpendicular distance of between 33-98 feet (10-30 meters) from the main roadway. Addresses included in this count have a value of AP09 in the **LocationCode** output field.
- **Street Access Point**—The geocode represents the primary point of access from the street network. This address point type is located where the driveway or other access road intersects the main roadway. Addresses included in this count have a value of AP10 in the **LocationCode** output field.

- **Base Parcel Point**—The Centrus point data includes individual parcels that may be "stacked". These stacked parcels are individually identified by their unit or suite number, and GeocodeUSAddress is able to match to this unit number and return the correct APN. Addresses included in this count have a value of AP21 in the **LocationCode** output field.
- **Backfill Address Point**—The precise parcel centroid is unknown. The address location assigned is based on two known parcel centroids. Addresses included in this count have a value of AP22 in the **LocationCode** output field.
- **Virtual Address Point**—The precise parcel centroid is unknown. The address location assigned is relative to a known parcel centroid and a street segment end point. Addresses included in this count have a value of AP23 in the **LocationCode** output field.
- **Interpolated Address Point**—The precise parcel centroid is unknown. The address location assigned is based on street segment end points. Addresses included in this count have a value of AP24 in the **LocationCode** output field.
- **Unknown Point-Level Geocode**—The type of geocode is not known.

ZIP Centroid Matching Details

This section describes the types of ZIP Code centroids and census centroids returned by GeocodeUSAddress.

- **Location Accuracy**—These fields describe the accuracy of the ZIP Code centroids.
- **ZIP + 4**—The centroid indicates the center of a ZIP + 4 code. This is the most accurate type of ZIP centroid. Addresses included in this count have a value of 9 in the third character of the value in the **LocationCode** output field.
- **ZIP + 2**—The centroid represents the center of a ZIP + 2 code. Addresses included in this count have a value of 7 in the third character of the value in the **LocationCode** output field.
- **ZIP Code**—The centroid represents the center of a five-digit ZIP Code. This is the least accurate type of ZIP centroid. Addresses included in this count have a value of 5 in the third character of the value in the **LocationCode** output field.
- **Census Accuracy**—These fields describe the accuracy of the census centroids.
- **Block Group**—The centroid represents the center of a block group. This is the most accurate type of census centroid. Addresses included in this count have a value of that begins with ZB in the **LocationCode** output field.
- **Census Tract**—The centroid represents the center of a census tract. Addresses included in this count have a value of that begins with ZT in the **LocationCode** output field.
- **County**—The centroid represents the center of a county. This is the least accurate type of census centroid. Addresses included in this count have a value of that begins with ZC in the **LocationCode** output field.

LACS/Link Statistics

This section describes the results of LACS/Link address processing. For information on LACS/Link, see **Locatable Address Conversion System (LACS)** on page 211.

- **Records processed by LACS/Link**—Addresses that were processed using LACS/Link.
- **LACS/Link Matched**—Addresses that were matched to addresses in the LACS/Link database.
- **LACS/Link Matched w/ dropped unit info**—Addresses whose secondary address information was removed in order to obtain a LACS^{Link} match.
- **Not LACS/Link Matched**—Addresses that GeocodeUSAddress attempted to match to LACS^{Link} but were not found in the LACS^{Link} database.
- **Not LACS/Link Converted**—The address matched a LACS^{Link} record but was not converted.
- **Not LACS/Link Converted - highrise default**—The address matched a highrise default record but was not converted.
- **Last LACS/Link false positive record**—This is the record number within the input file of the last address to result in a false positive. For example, if the 5th record in the file was a LACS^{Link} false

positive, this field would contain "5". For more information on false positives, see **Encountering False Positives** on page 351.

Delivery Point Validation

This section describes the results of DPV address processing. For information on DPV, see **Delivery Point Validation (DPV)** on page 211.

- **Records processed by DPV**—The number of addresses that were processed using DPV.
- **DPV Records with ZIP + 4**—Addresses that contained a ZIP + 4 code and were processed by DPV.
- **DPV Confirmed**—The number of addresses that were verified as deliverable addresses.
- **Primary Confirmed, Secondary Missing**—The primary portion of the address (the house number and street) was verified. The address requires a secondary element (for example, a suite or apartment number) to be a deliverable address, and the secondary information was missing from the input address.
- **Primary Confirmed, Secondary Incorrect**—The primary portion of the address (the house number and street) were verified. The address requires a secondary element (for example, a suite or apartment number) to be a deliverable address, and the secondary information in the input address was incorrect.
- **DPV CMRA Confirmed**—Commercial Mail Receiving Agency (CMRA) addresses confirmed by DPV.
- **DPV Not Confirmed**—Addresses that could not be verified as deliverable.
- **USPS Street Records Confirmed**—Street addresses that were confirmed by DPV.
- **USPS General Delivery Records Confirmed**—DPV processing confirmed that the address accepts general delivery mail.
- **Records with confirmed CMRA**—Commercial Mail Receiving Agency (CMRA) addresses that were confirmed with DPV.
- **Records not confirmed CMRA**—Commercial Mail Receiving Agency (CMRA) addresses that could not be confirmed with DPV.
- **DPV False Positive Seed table hits**—Addresses that matched to DPV false positive records. For more information, see **Encountering False Positives** on page 351.

Records with DPV Footnote

This section lists the DPV footnote codes that were returned for the job. For an explanation of the DPV footnote codes, see **DPV** on page 271.

USPS Firm Records

This section describes the results of address validation for firm (business) addresses.

- **Confirmed**—GeocodeUSAddress confirmed that the address is a business address.
- **Confirmed with PMB presented**—GeocodeUSAddress confirmed that the address is a business address, and the business address contains a private mailbox (PMB).
- **Failed primary house number**—Business addresses that contained a primary house number which could not be confirmed.
- **Failed secondary unit number**—Business addresses that contained a secondary unit number which could not be confirmed.

USPS Highrise Records

This section describes the results of DPV processing for highrise addresses.

- **Confirmed**—Highrise addresses that were confirmed by DPV.
- **Confirmed with PMB presented**—Highrise addresses that contain a Private Mailbox (PMB) and were confirmed by DPV.
- **Conf. CMRA with/without PMB**—Highrise addresses that are also CMRA addresses, and that did or did not contain a Private Mailbox (PMB) address element.

- **Failed primary house number**—Highrise addresses that contained a primary house number which could not be confirmed.
- **Failed secondary unit number**—Highrise addresses that contained a secondary unit number which could not be confirmed.

USPS PO Box Records

This section describes the results of DPV processing for PO box addresses.

- **Confirmed**—PO Box addresses that were confirmed by DPV.
- **Failed primary box number**—PO Box addresses that contained a primary box number which could not be confirmed.

USPS Rural Route Records

This section describes the results of DPV processing for rural route addresses.

- **Confirmed**—Rural Route addresses that were confirmed by DPV.
- **Conf. CMRA with/without PMB**—Rural Route addresses that were also CMRA addresses, and that did or did not contain a Private Mailbox (PMB) address element.
- **PMB Presented**—Rural Route addresses that contained a Private Mailbox (PMB) address element.
- **Failed primary house number**—Rural Route addresses that contained a primary house number which could not be validated.

USPS Street Records

This section describes the results of DPV processing for street addresses.

- **Confirmed**—Street addresses that were confirmed by DPV.
- **Confirmed with PMB presented**—Street addresses that contained a Private Mailbox (PMB) and were confirmed by DPV.
- **Conf. CMRA with/without PMB**—Street addresses that were also CMRA addresses, and that did or did not contain a Private Mailbox (PMB) address element.
- **Failed primary house number**—Street addresses that contained a primary house number which could not be confirmed.
- **Failed secondary unit number**—Street addresses that contained a secondary unit number which could not be confirmed.

GNAFPIDLocationSearch

GNAFPIDLocationSearch identifies the address and latitude/longitude coordinates for a Geocoded National Address File Persistent Identifier (G-NAF PID). The G-NAF PID is a 14-character alphanumeric string that uniquely identifies each G-NAF address in the G-NAF database (a database of Australian locations). The PID is constructed from a combination of the major address fields of the G-NAF database. An example of a G-NAF PID is:

GAVIC411711441

Note: You must have the G-NAF database installed to use GNAFPIDLocationSearch.

GNAFPIDLocationSearch is part of the Enterprise Geocoding Module. For more information on the Enterprise Geocoding Module, including a listing of other components included with it, see **Enterprise Geocoding Module** on page 204.

Input

GNAFPIDLocationSearch takes a G-NAF PID as input. The following table provides information on the format and layout of the input.

Note: Specify input using the DataTable class. For information on the DataTable class, see the "API Fundamentals" section.

Table 69: GNAFPIDLocationSearch Input

columnName	Format	Description
GNAFPID	String	The 14-character G-NAF persistent identifier you want to look up. For example: GAVIC411711441

Options

Geocoding Options

Table 70: GNAFPIDLocationSearch Geocoding Options

optionName	Description
GNAFPointType	Specifies whether to return the parcel latitude/longitude or the street frontage latitude/longitude. This option is only available if you have the G-NAF database installed. This option only affects addresses matched to the G-NAF database. One of the following: **P** In a street address match, return the exact location of the parcel. This is the standard G-NAF point which is the exact authoritative point returned by the G-NAF database. Default. **S** In a street address match, return the street frontage point for the parcel. The street frontage point is 12.5 metres from the front boundary of the parcel. Street frontage points are more suitable for routing applications.
Return8DecimalPlaceLatLong	Specifies whether to return the original latitude and longitude, precise up to eight digits after the decimal. This is the latitude/longitude that the candidate matched to in the G-NAF database. These are the original coordinates directly from the G-NAF data prior to truncation or rounding. This option is only available if you have the G-NAF database installed. This option only affects addresses matched to the G-NAF database. **Y** Yes, return the original latitude/longitude, up to eight digits after the decimal. **N** No, do not return the original latitude/longitude.

Data Options

Table 71: GNAFPIDLocationSearch Data Options

optionName	Description
Database	Specifies the database to use to look up the parcel. Use the database name specified in the Management Console's Database Resources tool. For more information, see the *Spectrum™ Technology Platform Administration Guide*. **Note:** Only database resources that contain G-NAF databases are available in this list.

Output

Address Output

Table 72: Address Output

columnName	Description
AddressLine1	First line of the address.
AddressLine2	Second line of the address.
ApartmentLabel	The type of unit, such as apartment, suite, or lot.
ApartmentNumber	Unit number.
City	Municipality name.
Country	The three-letter ISO 3166-1 Alpha 3 country code.
County	The Local Government Authority (LGA).
FirmName	Name of the company or a place name.
HouseNumber	Building number for the matched location.
HouseNumberHigh	The highest house number of the range in which the address resides.
HouseNumberLow	The lowest house number of the range in which the address resides.
HouseNumberParity	Indicates if the house number range contains even or odd numbers or both. **E** Even **O** Odd **B** Both
LastLine	Complete last address line (city, state/province, and postal code).
LeadingDirectional	Street directional that precedes the street name. For example, the N in 138 N Main Street.
Locality	Generally a locality in rural areas or suburb in urban areas.
NumberOfCandidateRanges	Indicates whether or not the address has a house number. One of the following:

columnName	Description
	0 The address has no house number. Examples of addresses that have no house number are P.O. box addresses and general delivery addresses. **1** The address has a house number. For information about the range in which the house number falls, see the HouseNumberHigh, HouseNumberLow, and HouseNumberParity fields.
NumberOfRangeUnits	Indicates whether or not the address has a unit number, such as a suite number or apartment number. One of the following: **0** The address has no unit number. **1** The address has a unit number. For information about the range in which the unit number falls, see the UnitNumberHigh and UnitNumberLow.
PostalCode	The postcode for the address. The format of the postcode varies by country.
PostalCode.Addon	The second part of a postcode. For example, for Canadian addresses this will be the LDU. This field is not used by most countries.
PreAddress	Miscellaneous information that appears before the street name.
PrivateMailbox	This field is not currently used.
SegmentParity	Indicates which side of the street has odd numbers. **L** Left side of the street **R** Right side of the street **B** Both sides of the street **U** Undetermined
StateProvince	The state name.
StreetDataType	The default search order rank of the database used to geocode the address. A value of "1" indicates that the database is first in the default search order, "2" indicates that the database is second in the default search order, and so on. The default database search order is specified in the Management Console with the Database Resources tool.
StreetName	The street name.
StreetPrefix	The type of street when the street type appears before the base street name. For example, AVENUE: 12 **AVENUE** B KALGOORLIE WA 6430
StreetSuffix	The street type of the matched location. For example, AVE for Avenue.
TrailingDirectional	Street directional that follows the street name. For example, the N in 456 Washington N.
UnitNumberHigh	The highest unit number of the range in which the unit resides.
UnitNumberLow	The lowest unit number of the range in which the unit resides.

Geocode Output

Table 73: Geocode Output

columnName	Description
CoordinateSystem	The coordinate system used to determine the latitude and longitude coordinates. A coordinate system specifies a map projection, coordinate units, etc. An example is EPSG:4326. EPSG stands for European Petroleum Survey Group.
Latitude	Seven-digit number in degrees and calculated to four decimal places (in the format specified).
Longitude	Seven-digit number in degrees and calculated to four decimal places (in the format specified).

Result Codes

Result codes contain information about the success or failure of the geocoding attempt, as well as information about the accuracy of the geocode.

Table 74: Result Code Output

columnName	Description
Geocoder.MatchCode	Indicates how closely the input address matches the candidate address.
IsCloseMatch	Indicates whether or not the address is considered a close match. An address is considered close based on the "Close match criteria" options on the Matching tab. **Y** Yes, the address is a close match. **N** No, the address is not a close match.
MultiMatchCount	For street address geocoding, the number of matching address positions found for the specified address. For intersection geocoding, the number of matching street intersection positions found for the specified addresses.
Status	Reports the success or failure of the match attempt **null** Success **F** Failure
Status.Code	If the geocoder could not process the address, this field will show the reason. • Internal System Error • No Geocode Found • Insufficient Input Data • Multiple Matches Found • Exception occurred • Unable to initialize Geocoder

columnName	Description
	• No Match Found
Status.Description	If the geocoder could not process the address, this field will show a description of the failure. **Problem + explanation** — Returned when Status.Code = Internal System Error. **Geocoding Failed** — Returned when Status.Code = No Geocode Found. **No location returned** — Returned when Status.Code = No Geocode Found. **No Candidates Returned** — The geocoder could not identify any candidate matches for the address. **Multiple Candidates Returned and Keep Multiple Matches not selected** — The address resulted in multiple candidates. In order for the candidate address to be returned, you must specify `KeepMultimatch=Y`.
LocationPrecision	A code describing the precision of the geocode. One of the following: **0** — No coordinate information is available for this candidate address. **1** — Interpolated street address. **2** — Street segment midpoint. **3** — Postal code 1 centroid. **4** — Partial postal code 2 centroid. **5** — Postal code 2 centroid. **6** — Intersection. **7** — Point of interest. This is a placeholder value. Spectrum databases do not have POI data, so it is not possible to get this return. **8** — State/province centroid. **9** — County centroid. **10** — City centroid. **11** — Locality centroid. **12 - 15 (LocationPrecision codes)** — For most countries, LocationPrecision codes 12 through 15 are reserved for unspecified custom items. **13** — Additional point precision for unspecified custom item. **14** — Additional point precision for unspecified custom item. **15** — Additional point precision for unspecified custom item. **16** — The result is an address point.

columnName	Description	
	17	The result was generated by using address point data to modify the candidates segment data.
	18	The result is an address point that was projected using the centerline offset feature. You must have both a point and a street range database to use the centerline offset feature, and thereby return LocationPrecision 18.
StreetDataType	The default search order rank of the database used to geocode the address. A value of "1" indicates that the database is first in the default search order, "2" indicates that the database is second in the default search order, and so on. The default database search order is specified in the Management Console with the Database Resources tool.	

G-NAF Output

The following table lists output fields that are unique to the Australian Geocoded National Address File (G-NAF®) database. G-NAF is an optional database that is available for all six states and two territories. G-NAF is the only authoritative Australian national index of locality, street and number, validated with geographic coordinates.

Table 75: Austraila G-NAF Output

columnName	Description	
AUS.GNAF_CONFIDENCE	A number indicating how many G-NAF datasets the address is found in. A higher confidence level means that the same address was found in more data contributor sources. One of the following:	
	<number>	The number of datasets the address was found in, minus 1. For example, a value of 0 indicates that the address was found in one contributor's dataset, a value of 1 indicates that the address was found in two contributors' datasets, a value of 2 indicates that the address was found in three contributors' datasets, and so forth.
	-1	The address could not be found in any G-NAF dataset.
AUS.GNAF_EIGHT_DECIMAL_PLACE_LATITUDE	The parcel latitude, precise to eight digits after the decimal. This is the latitude that the candidate matched to in the G-NAF database. These are the original coordinates directly from the G-NAF data prior to truncation or rounding. This field is only returned if you specify **Return8DecimalPlaceLatLong=Y**.	

columnName	Description
AUS.GNAF_EIGHT_DECIMAL_PLACE_LONGITUDE	The parcel longitude, precise to eight digits after the decimal. This is the longitude that the candidate matched to in the G-NAF database. These are the original coordinates directly from the G-NAF data prior to truncation or rounding. This field is only returned if you specify **Return8DecimalPlaceLatLong=Y**.
AUS.GNAF_GEOCODE_LEVEL	A number indicating the level o f geocode for the address. Every principal address within the G-NAF database has at least a locality level geocode. They may also have a street level geocode and a point level geocode. One of the following: **0** No geocode. **1** Parcel level geocode only (no locality or street level geocode). **2** Street level geocode only (no locality or parcel level geocode). **3** Street and parcel level geocodes (no locality geocode). **4** Locality level geocode only (no street or parcel level geocode). **5** Locality and parcel level geocodes (no street level geocode). **6** Locality and street level geocodes (no parcel level geocodes). **7** Locality, street and parcel level geocodes.
AUS.GNAF_PID	The G-NAF Persistent Identifier (G-NAF PID) is a 14-character alphanumeric string that uniquely identifies each G-NAF address. The PID is constructed from a combination of the major address fields of the G-NAF Dictionary. An example of a G-NAF PID is: GAVIC411711441
AUS.GNAF_RELIABILITY	A number indicating the geocode precision. Reliability is related to the dictionary used to determine the geocode. Data with geocoded reliability levels 1, 2, and 3 is contained in the GNAF123 Dictionary. This is point (parcel) level geocoded data. Data with geocoded reliability levels 4, 5, and 6 is contained in the GNAF456 Dictionary. This contains non-parcel centroid geocoded data. **1** Geocode accuracy recorded to appropriate surveying standard. For example, this could apply to an address level geocode that was manually geocoded. Geocode resolution is

columnName	Description
	sufficient to place the centroid within address site boundary with a GPS. **2** Geocode accuracy sufficient to place centroid within address site boundary. For example, this could apply to an address level geocode that was automatically calculated as the centroid of the corresponding cadastre parcel. **3** Geocode accuracy sufficient to place centroid near (or possibly within) address site boundary. For example, this could apply to an address level geocode that was automatically calculated by calculating where on the road the address was likely to appear based upon other bounding geocoded addresses. **4** Geocode accuracy sufficient to associate address site with a unique road feature. For example, this could apply to a street level geocode that was automatically calculated by using the road centerline reference data. **5** Geocode resolution sufficient to associate address site with a unique locality or neighborhood. For example, this could apply to a locality level geocode that was automatically calculated as the centroid of the locality. **6** Geocode resolution sufficient to associate address site with a unique region. For example, this could apply to a locality level geocode that was derived from topographic feature.
AUS.LEVEL_NUMBER	The number of a floor or level in a multi-story building. For example, **Floor 2** , 17 Jones Street The G-NAF database includes level information for some Australian states. Level information may be associated with unit information, but not necessarily. If the G-NAF database contains multiple records with the same level, the level information is returned only if the input address contains unique content (such as a unit number). If the G-NAF dictionary has level information for an address, that information is returned with the matched candidate. The correct level information is returned (when available) even if the input address did not include level information, or if the input had the wrong level information. If the input address has level information but the G-NAF database does not include level information for the matching address,

columnName	Description
	then the input level information is discarded since it is not validated in the G-NAF data.
AUS.LEVEL_TYPE	The label used for a floor of a multi-story building. For example, "Level" or "Floor". In this example, the level type is "Level": Suite 3 **Level 7**, 17 Jones Street In this example, Suite 3 is a unit. The G-NAF database includes level information for some Australian states. Level information may be associated with unit information, but not necessarily. If the G-NAF database contains multiple records with the same level, the level information is returned only if the input address contains unique content (such as a unit number). If the G-NAF dictionary has level information for an address, that information is returned with the matched candidate. The correct level information is returned (when available) even if the input address did not include level information, or if the input had the wrong level information. If the input address has level information but the G-NAF database does not include level information for the matching address, then the input level information is discarded since it is not validated in the G-NAF data.
AUS.MESH_BLOCK_ID	A Meshblock is the smallest geographic unit for which statistical data is collected by the Australian Bureau of Statistics (ABS). Meshblocks usually contain a minimum of 20 to 50 households. This is about one fifth the size of a Collection District (CD). You can use the Meshblock ID to do additional attributions against your own data.
AUS.LOT_NUMBER	Lot numbers are returned for G-NAF candidates because some rural addresses do not have adequate physical or house number information.
AUS.GEOCONTAINMENT	This specifies whether the returned coordinates are inside or outside the address boundary. Values are YES for coordinates within, or NO for coordinates outside the boundary. Candidates with a G-NAF Reliability Level of 2 are considered within the address site boundary. Candidates with a G-NAF Reliability Level of 3 are generally considered within the address site boundary, although exceptions may exist.
AUS.GEOFEATURE	This field returns a geocode feature type if that was not provided in other G-NAF fields. GEOFEATURE corresponds to Geocode Types (GEOCODE_TYPE_AUT Codes) that are described

columnName	Description
	in the PSMA Data Product Description Version 2.7 (Aug. 2012).
AUS.STREET_TYPE_ABB	This is a PSMA standard abbreviation for the street type. This G-NAF field is provided for National Address Management Framework (NAMF) compliance.

ReverseAPNLookup

ReverseAPNLookup allows you to look up an address using:

- An Assessor's Parcel Number (APN). An APN is an ID number assigned to a piece of land by a county assessor. An APN is unique only within a county.
- A FIPS county code. A Federal Information Processing Standard (FIPS) code is an ID number assigned to a county by the U.S. Federal government.
- A FIPS state code. A FIPS state code is an ID number assigned to each state by the U.S. Federal government.

These three pieces of information, used together, can uniquely identify a specific parcel. You must use all three pieces of information to perform a lookup using ReverseAPNLookup.

Note: ReverseAPNLookup only works for U.S. addresses for which APN data is available. See the coverage map included with the points database for more information.

ReverseAPNLookup is part of the Enterprise Geocoding Module. For more information on the Enterprise Geocoding Module, including a listing of other components included with it, see **Enterprise Geocoding Module** on page 204.

Input

ReverseAPNLookup takes an APN, FIPS county code, and FIPS state code as input. The following table provides information on the format and layout of the input.

Table 76: Reverse APNLookup Input Data

columnName	Format	Description
APN	String [45]	The assessor's parcel number (APN) for the property you want to look up.
InputKeyValue	String	User-defined data, such as a record ID or source code.
USFIPSCountyNumber	String [5]	The FIPS county code for the county in which the property resides.
USFIPSStateCode	String [2]	The FIPS state code for the state in which the property resides.

Options

The following table lists the options that control ReverseAPNLookup processing.

Table 77: ReverseAPNLookup Options

optionName	Description
Dataset	Specifies the database to use to look up the parcel. Use the database name specified in the Management Console's Database Resources tool. For more information, see the *Spectrum™ Technology Platform Administration Guide*.
LatLongFormat	Specifies the format for returned latitude/longitude. **Decimal** (90.000000-180.000000) Default. **Integer** (90000000-180000000)
RetrieveElevation	Specifies whether or not ReverseAPNLookup returns the elevation of the address. Elevation is the distance above or below sea level of a given location. The elevation is returned in the **Elevation** output field, which is part of the Latitude/Longitude output group. **Note:** This option requires that you have licensed and installed the Centrus Premium Points database. Elevation data is not available for all addresses. See the coverage map included with the points database.
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345
OutputVerbose	Specifies whether or not ReverseAPNLookup provides an additional description field as output. This field provides the text equivalent to a field represented by a code. For example, LocationCode returns a code that indicates the accuracy (quality) of the assigned geocode. LocationCode.Description provides the description for the code returned. **Y** Include verbose fields. **N** Do not include verbose fields. Default.
OutputRecordType	Specifies optional data to include in the output. Note that ReverseAPNLookup always returns the default data listed in **Default Output** on page 301. The data you select here is returned with the default output data. • **C**—Census • **L**—Latitude/Longitude • **E**—Parsed Elements • **Q**—Qualifiers • **R**—Range • **S**—Segment For a list of fields included in each record type, see **Output** on page 300.

optionName	Description
	If you do not want all of the fields in a group returned, do not select the group, and instead list only those fields you want returned in `OutputFields`.
OutputFields	Indicate the individual output fields you want returned. List fields with a pipe (|) between each field. You can use this field instead of the `OutputRecordType` field to limit the output to the specific fields you want. Default list: AddressLine1|LastLine|Longitude|Latitude|MatchCode|LocationCode

Output

ReverseAPNLookup returns the following categories of data:

- **Census** on page 300
- **Default Output** on page 301
- **Latitude/Longitude** on page 303
- **Parsed Elements** on page 304
- **Qualifiers** on page 305
- **Range** on page 307
- **Segment** on page 308

Census

The Census output fields contain census information from the U.S. 2000 Census. To include census data in the output, set **OutputRecordType** = C.

Table 78: Census Output Fields

columnName	Max. Field Length (bytes)	Description
BlockSuffix	2	Single character block suffix for split Census blocks. Returns A or B. Only available in Centrus Enhanced data.
CBSACode	6	Indicates Core Based Statistical Area (CBSA).
CBSADivisionCode	6	Denotes a subdivision of a CBSA.
CBSADivisionName	128	Describes a subdivision of a CBSA. Only returned when you set the configuration options to return additional descriptions (verbose).
CBSAMetro	2	Metropolitan Statistical Area. Valid values include: **Y** Metro statistical area **N** Micro statistical area **null** Data unavailable

columnName	Max. Field Length (bytes)	Description
CBSAName	128	Describes CBSA. Only returned when you set the configuration options to return additional descriptions (verbose).
CensusBlockID	16	The ID of the Census Federal Information Processing Standard (FIPS) code.
CensusTract	7	Six digits extracted from the CensusBlockID.
CSACode	4	Denotes the code for a geographic entity that consists of 2 or more adjacent CBSAs with employment interchange measures of at least 15.
CSAName	128	Describes the name for a geographic entity that consists of 2 or more adjacent CBSAs with employment interchange measures of at least 15. Only returned when you set the configuration options to return additional descriptions (verbose).
USCountyName	128	Name of the county, including the text "County" or "Parish." Only returned when you set the configuration options to return additional descriptions (verbose).
USFIPSStateCountyCode	6	Five-digit FIPS code for state and county extracted from the CensusBlockID.

Default Output

ReverseAPNLookup always returns the address, geocode, and result indicators.

Table 79: Default Output Fields

columnName	Max. Field Length (bytes)	Description
AdditionalInputData	61	This field is populated with information from the mail stop output field. **Note:** ReverseAPNLookup does not process this information. It simply includes the information as entered in the input data.
AddressLine1	104	First line of the address.
AddressLine2	104	Second line of the address.
APN	46	The Assessor's Parcel Number that was specified in the input.
City	29	Municipality name.
Confidence	4	Indicates the confidence in the output provided. The range is from 0 (zero) to 100, with 0 being no match and 100 being an exact match.

columnName	Max. Field Length (bytes)	Description
Country	25	The name of the country. Since ReverseAPNLookup only works for U.S. locations, this field will always contain **United States of America**.
Distance	8	The distance, in feet, from the input geocode to nearest matched address or intersection.
Elevation	11	The distance in feet above or below sea level of the parcel.
FirmName	41	Name of the company.
LastLine	61	Complete last address line (municipality, state, and postal code).
Latitude	11	Seven-digit number in degrees and calculated to 4 decimal places (in the format specified).
LocationCode	5	Indicates the accuracy (quality) of the assigned geocode. For more information, see **Address Location Codes** on page 328.
Longitude	12	Seven-digit number in degrees and calculated to 4 decimal places (in the format specified).
MatchCode	5	Indicates the portions of the address that matched to the directory file. For more information, see **Geocoding Match Codes** on page 341.
PercentGeocode	6	The percent along the street segment that matches the geocode. For example, if the returned geocode falls 1/3 along the way of the entire street segment, the percent is 33.000. **Note:** This value is always 0.0 for matches to point-level data and intersections.
PostalCode	10	Nine-digit ZIP Code with or without a hyphen.
PostalCode.AddOn	5	Four-digit ZIP Code extension.
PostalCode.Base	6	Five-digit ZIP Code.
ProcessedBy	4	The feature code for the stage that processed the request. The value is EnterpriseGeocoding for ReverseAPNLookup.
StateProvince	3	Two-character state abbreviation.
Status	2	Reports the success or failure of the match attempt **null** Success **F** Failure
Status.Code	23	Reason for failure: • Internal System Error • No Address Found

columnName	Max. Field Length (bytes)	Description
		• Insufficient Input Data
Status.Description	128	Description of the problem: **Problem + explanation** Returned when Status.Code = Internal System Error. **Geocoding Failed** Returned when Status.code = No Address Found) **No location returned** Returned when Status.code = No Address Found.
StreetDataType	20	The data set that ReverseAPNLookup attempted to match against. **USPS** USPS **TIGER** TIGER **TOMTOM** TomTom Streets data set **SANBORN POINT DATA** Sanborn Points data set **NAVTEQ** NAVTEQ Streets data set **TOMTOM POINT DATA** TomTom Points data set **AUXILIARY** Auxiliary file **CENTRUS POINT DATA** Centrus Points data set **USER DICTIONARY** User Dictionary **NAVTEQ POINT DATA** NAVTEQ Points data set **MASTER LOCATION** Master Location Data
StreetSide	2	Indicates the side of the street the range occupies. One of the following: **L** The range occupies the left side of the street. **R** The range occupies the right side of the street. **B** The range occupies both sides of the street. **U** Undetermined.
USFIPSCountyNumber	4	Three-digit FIPS county code specified in the input.
USFIPSStateCode	3	Two-digit FIPS state code specified in the input.
USUrbanName	6	USPS® urbanization name. Puerto Rican addresses only.

Latitude/Longitude

The latitude/longitude output fields contain the geographic coordinates of the location and elevation. To include latitude/longitude output fields in the output, set **OutputRecordType** = L.

Table 80: Latitude/Longitude Output Fields

columnName	Max. Field Length (bytes)	Description
Elevation	11	The distance in feet above or below sea level of the parcel.
Latitude	11	7-digit number in degrees and calculated to 4 decimal places (in the format specified).
Longitude	12	7-digit number in degrees and calculated to 4 decimal places (in the format specified).

Parsed Elements

The Parsed Elements output fields contain standard address information as individual units, such as street suffixes (AVE) and leading directionals (N and SE). To include parsed elements in the output, set **OutputRecordType** = E.

Table 81: Parsed Elements Output Fields

columnName	Max. Field Length (bytes)	Description
ApartmentLabel	5	Apartment designator (such as STE or APT), for example: 123 E Main St. APT 3
ApartmentLabel2	5	Secondary apartment designator, for example: 123 E Main St. APT 3, 4th Floor
ApartmentNumber	12	Apartment number, for example: 123 E Main St. APT 3
ApartmentNumber2	12	Secondary apartment number, for example: 123 E Main St. APT 3, 4th Floor
City	29	Municipality name.
CrossStreetLeadingDirectional	3	Leading directional, for example: 123 E Main St. Apt 3
CrossStreetName[2]	41	Cross street name, for example: 123 E Main St. Apt 3
CrossStreetSuffix	5	Cross street suffix, for example: 123 E Main St. Apt 3
CrossStreetTrailingDirectional	3	Cross street trailing directional, for example: 123 Pennsylvania Ave NW
HouseNumber	12	Building number, for example: 123 E Main St. Apt 3 **Note:** This is an approximate building number based on the APN, FIPS county code, and FIPS state code provided. This approximate address may not exist or may not accept mail delivery.

[2] ReverseAPNLookup only returns Cross street outputs if you entered an intersection as an address. For example, entering Pearl and 28th, Boulder, CO returns cross street information. Entering 2800 Pearl, Boulder, CO does NOT return cross street information.

columnName	Max. Field Length (bytes)	Description
LeadingDirectional	3	Leading directional, for example: 123 E Main St. Apt 3
PrivateMailbox	9	Private mailbox indicator. Not output for multiline input.
PrivateMailbox.Designator	5	The type of private mailbox. Possible values include: • Standard • Non-Standard
RRHC	2	Rural Route/Highway Contract indicator.
StreetName	41	Street name, for example: 123 E Main St. Apt 3
StreetSuffix	5	Street suffix, for example: 123 E Main St. Apt 3
TrailingDirectional	3	Trailing directional, for example: 123 Pennsylvania Ave NW

Qualifiers

The qualifiers output fields contain qualification information on the match, such as the location code and the match code. To include qualifier output fields in the output, set **OutputRecordType** = Q.

Table 82: Qualifiers Output Fields

columnName	Max. Field Length (bytes)	Description
CountryLevel	2	The category of postal data available. Always returns A in ReverseAPNLookup—Validates, corrects, and provides missing postal code, city name, state/county name, street address elements, and country name.
DatabaseVersion	15	USPS publish date, in the format Month Year.
EWSMatch	2	Indicates if ReverseAPNLookup denied a match because of Early Warning System (EWS) data. **Y** EWS denied a match. **null** EWS did not deny a match. For more information on EWS, see **Early Warning System (EWS)** on page 212.
ExpirationDate	11	Date the database expires, in the format MM/DD/YYYY.
GeoStanMatchScore	13	Record matching score (for multimatches only).
Intersection	2	Indicates if ReverseAPNLookup found a cross-street match. **T** True, a cross-street match was found. **F** False, a cross-street match was not found.

columnName	Max. Field Length (bytes)	Description
IsAlias	4	ReverseAPNLookup located a matched record by an index alias. Returns 3 characters. The first is an **N** for normal street match or **A** for alias match (including buildings, aliases, firms, etc.). The next 2 characters are: **01** Basic index (normal address match) **02** USPS street name alias index **03** USPS building index **04** USPS firm name index **05** Statewide intersection alias match (when using the Usw.gsi or Use.gsi file) **06** Spatial data street name alias (when using the Us_pw.gsi, Us_pe.gsi, Us_psw.gsi, or Us_pse.gsi file) **07** Alternate index (when using Zip9.gsu, Zip9e.gsu, and Zip9w.gsu) **08** LACS^{Link} **09** Auxiliary file match **10** Centrus Alias index (when using usca.gsi)
LACSAddress	2	Indicates if ReverseAPNLookup converted an address due to the Locatable Address Conversion System (LACS). **L** Converted **null** Not converted For more information on LACS, see **Locatable Address Conversion System (LACS)** on page 211.
LocationCode.Description	128	LocationCode converted to text. Only returned when you set the configuration options to return additional descriptions (verbose).
MatchCode.Description	128	MatchCode converted to text. Only returned when you set the configuration options to return additional descriptions (verbose).
RecordType	18	Indicates the record type: • GeneralDelivery • HighRise • FirmRecord • Normal • PostOfficeBox • RRHighwayContract
RecordType.Default	2	Indicates type of match that occurred for the record type HighRise or RRHighwayContract: **Y** Default match

columnName	Max. Field Length (bytes)	Description
		N Exact match **U** Not matched
StreetDataCode	3	Indicates the data used to obtain a match. **0** USPS data in either the Centrus Enhanced, Centrus TomTom, or Centrus NAVTEQ database. **1** TIGER data in the Centrus Enhanced database. **2** TomTom data in the Cenrus TomTom database. **6** NAVTEQ data in the Centrus NAVTEQ database. **7** TomTom point-level data in the Centrus TomTom Points database. **8** Point-level data from the Centrus Points database. **9** Auxiliary file data **10** User Dictionary. **11** Point-level data from NAVTEQ Points database. **12** Master Location Data (MLD). For more information about these databases, see **Enterprise Geocoding Databases** on page 205.

Range

The range output fields contain information on the street range, such as the high and low unit numbers. To include range data fields in the output, set **OutputRecordType** = R.

Table 83: Range Output Fields

columnName	Max. Field Length (bytes)	Description
Alternate	2	USPS code that specifies whether a record is a base or alternate record. **B** Base record. Base records can represent a range of addresses or an individual address, such as a firm record. **A** Alternate record. Alternate records are individual delivery points.
HouseNumberHigh	12	House number high.
HouseNumberLow	12	House number low.
HouseNumberParity	2	Indicates if the house number range contains even or odd numbers.

columnName	Max. Field Length (bytes)	Description
		E Even
		O Odd
		B Both
PostalCodeExtensionHigh	5	4-digit ZIP Code extension high.
PostalCodeExtensionLow	5	4-digit Zip Code extension low.
UnitNumberHigh	12	Unit number high.
UnitNumberLow	12	Unit number low.
UnitNumberParity	2	Indicates if the unit number range contains even or odd numbers.
		E Even
		O Odd
		B Both

Segment

The segment output fields contain information on the street segment identified by the data provider. To include segment data fields in the output, set **OutputRecordType** = S.

Table 84: Segment Output Fields

columnName	Max. Field Length (bytes)	Description
BlockLeft	16	Provides the Census FIPS Code that indicates the address is on the left side of the street.
BlockRight	16	Provides the Census FIPS Code that indicates the address is on the right side of the street.
BlockSuffixLeft	2	Current left Block suffix for Census 2000 Geography. Returns A or B. Only available in Centrus Enhanced data.
BlockSuffixRight	2	Current right Block suffix for Census 2000 Geography. Returns A or B. Only available in Centrus Enhanced data.
RoadClass	3	The type of road: **1** The road is a major road. **2** The road is a minor road.

columnName	Max. Field Length (bytes)	Description
PointCode	11	Unique point ID assigned by the data provider. This field is blank if the matched record is not from point-level data.
SegmentCode	11	Unique 10-digit segment ID assigned by the street network provider.
SegmentDirection	2	Indicates the order of numbers on a segment. **F** Forward **R** Reversed **B** Both **U** Undetermined
SegmentHouseNumberHigh	12	A high range number in the segment.
SegmentHouseNumberLow	12	A low range number in the segment.
SegmentLength		The length, in feet, of a block segment. *This field is deprecated.*
SegmentParity	2	Indicates which side of the street has odd numbers. **L** Left side of the street **R** Right side of the street **B** Both sides of the street **U** Undetermined

Reverse Geocode Address Global

For information on using the API to access Reverse Geocode Address Global, see the geocoding guides.

ReverseGeocodeUSLocation

ReverseGeocodeUSLocation takes a latitude and longitude point as input and returns the address that is the best match for that point. For example, you could enter the following information:

Longitude: -105239771 Latitude: 40018912 Search Distance: 150 feet

This input would result in the following output:

4750 WALNUT ST BOULDER, CO 80301-2538
MatchCode = NS0
LocCode = AS0
Lon = -105239773
Lat = 40018911
Distances:
Search = 150
Offset = 50
Squeeze = 50
Nearest = 50.0

Pct Geocode = 94.0
SegID = 472881795
PtID = GDT
Block = 080130122032066
County Name = BOULDER COUNTY
DPBC = 50

Note: The address returned is an approximate address based on the latitude and longitude provided. This approximate address may not exist or may not accept mail delivery.

ReverseGeocodeUSLocation is part of the Enterprise Geocoding Module. For more information on the Enterprise Geocoding Module, including a listing of other components included with it, see **Enterprise Geocoding Module** on page 204.

ReverseGeocodeUSLocation processes geocodes in the following order:

1. ReverseGeocodeUSLocation defines a small rectangle based on your input geocode and search distance.
2. ReverseGeocodeUSLocation computes the distance between each street segment and the input location.
3. If one segment is closest, ReverseGeocodeUSLocation finds the offset and interpolated percentage (using the squeeze factor) and the street side. It then computes an approximate house number based on this information.

   If there is more than one segment that is equally close to the input location, a multi-match occurs. ReverseGeocodeUSLocation returns the information for all of the equally close segments so that you can determine which segment is applicable.
4. ReverseGeocodeUSLocation returns the address information, including the segment range, the approximate house number, and the parity of the range along with other standard address information.

Note: Although many of the standard address matching outputs apply to the reverse geocoding option, several outputs are unavailable (such as LACS^{Link} information and unit numbers). ReverseGeocodeUSLocation returns these outputs as blank. ReverseGeocodeUSLocation also has outputs specific to reverse geocode processing, such as specific match codes and the distance from the input location to the matched segment.

To use ReverseGeocodeUSLocation, you need additional data files, called GSX files. There is an option to install these files when you install the geocoding database. The GSX files must be installed the GSX subdirectory of the geocoding database. If you install the Centrus Enhanced Points, Centrus Premium Points, or Centrus TomTom Points database, you must recreate the GSX files. Consult with Pitney Bowes Software Technical Support if you need more information on GSX files.

Input

ReverseGeocodeUSLocation takes longitude and latitude information as input. The following table provides information on the format and layout of the input.

Table 85: ReverseGeocodeUSLocation Input Data

columnName	Format	Description
Latitude	String	Latitude of the point for which you want address information returned. Specify latitude in millionths of decimal degrees.
Longitude	String	Longitude of the point for which you want address information returned. Specify longitude in millionths of decimal degrees.

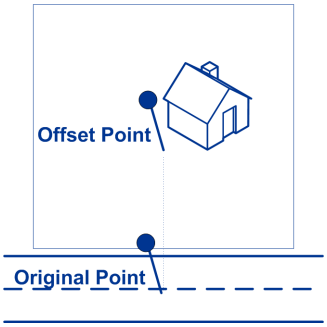
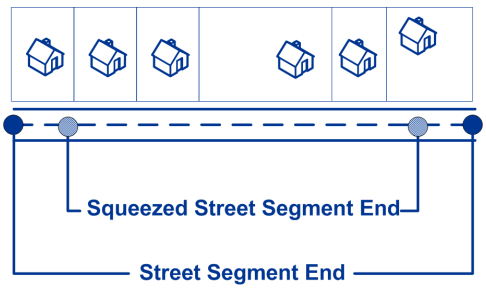
Options

Configuration Options

The following table lists the configuration options for ReverseGeocodeUSLocation.

Table 86: ReverseGeocodeUSLocation Configuration Options

optionName	Description
Dataset	The name of the database that contains the data to use in the search process. Use the database name specified in the Management Console's Database Resources tool. **Note:** To use the reverse geocoding service, you must install a data set that includes .gsx files. For more information, see the *Spectrum™ Technology Platform Administration Guide*.
SearchDistance	Specifies the radius, in feet, that ReverseGeocodeUSLocation search for matches. The range is 0 - 5280 feet, with a default value of 150 feet.
FindClosestPoint	Enables matching to the nearest point address within the search radius, rather than to the closest feature (e.g. street segment or intersection as well as point addresses). **Note:** This option requires that at least one streets data set and one points data set are loaded; otherwise, the match will be made to the closest feature. **Y** Match to the nearest point address within the search radius. **N** Match to the closest point address or feature. Default.
FindNearestAddress	Specifies whether or not ReverseGeocodeUSLocation should find the nearest interpolated address to the input geocode. **Y** Find the nearest address. Default. **N** Do not find the nearest address. **Note:** You can use this option with the FindNearestIntersection option to geocode to both addresses and intersections.
FindNearestUnranged	Specifies whether or not ReverseGeocodeUSLocation can match to a street segment that does not have a number range. This option is active when FindNearestAddress=Y. **Y** Allow ReverGeocodeUSLocation to match to an unranged street segment. Default. **N** Do not allow ReverseGeocodeUSLocation to match to an unranged street segment. **Note:** If you are using the point-level data option, ReverseGeocodeUSLocation ignores the Nearest Unranged option.

optionName	Description
FindNearestIntersection	Specifies whether or not ReverseGeocodeUSLocation should find the nearest street intersection to the input geocode. **Y** Find the nearest street intersection. Default. **N** Do not find the nearest street intersection. **Note:** You can use this option with the FindNearestAddress option to geocode to both addresses and intersections.
Offset	Specifies the offset distance from the street segments. The range is 0 - 5280 feet, with the default value of 50 feet. The offset distance is used in street-level geocoding to prevent the geocode from being in the middle of a street. It compensates for the fact that street-level geocoding returns a latitude and longitude point in the center of the street where the address is located. Since the building represented by an address is not on the street itself, you do not want the geocode for an address to be a point on the street. Instead, you want the geocode to represent the location of the building which sits next to the street. For example, an offset of 50 feet means that the geocode will represent a point 50 feet back from the center of the street. The distance is calculated perpendicular to the portion of the street segment for the address. Offset is also used to prevent addresses across the street from each other from being given the same point. The following diagram shows an offset point in relation to the original point.  Street coordinates are accurate to 1/10,000th of a degree and interpolated points are accurate to 1/1,000,000th of a degree.
Squeeze	Specifies the distance, in feet, to squeeze the street end points in street-level geocoding. The range is 0 -2147483647 feet, with the default value of 50 feet. The following diagram compares the end points of a street to squeezed end points. 

optionName	Description
LatLonFormat	Specifies the format to use for returned latitude/longitude. **Decimal** The format is 90.000000-180.000000. Default. **Integer** The format is 90000000-180000000.
InputLatLonFormat	Specifies the format to use for input latitude/longitude. **Decimal** The format is 90.000000-180.000000. Default. **Integer** The format is 90000000-180000000.
RetrieveElevation	Specifies whether or not ReverseGeocodeUSLocation returns the elevation of the address. Elevation is the distance above or below sea level of a given location. The elevation is returned in the **Elevation** output field, which is part of the Latitude/Longitude output group. **Note:** This option requires that you have licensed and installed the Centrus Premium Points database. Elevation data is not available for all addresses. See the coverage map included with the points database.
RetrieveAPN	Specifies whether or not ReverseGeocodeUSLocation should determine the address's APN (assessor's parcel number). The APN is an ID number assigned to a property by the local property tax authority. The APN is returned in the **APN** output field, which is part of the Census output group. **Note:** This option requires that you have licensed and installed the Cenrus Enhanced Points or Centrus Premium Points database. APN data is not available for all addresses. See the coverage map included with the points database.

Output Format

The following table lists the options that control the format of the output .

Table 87: ReverseGeocodeUSLocation Output Format Options

optionName	Description
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345

optionName	Description
OutputVerbose	Specifies whether or not ReverseGeocodeUSLocation provides an additional description field as output. This field provides the text equivalent to a field represented by a code. For example, LocationCode returns a code that indicates the accuracy (quality) of the assigned geocode. LocationCode.Description provides the description for the code returned. **Y** Include verbose fields. **N** Do not include verbose fields. Default.

Output Data

The following table lists the options that control which data is returned by ReverseGeocodeUSLocation.

Table 88: ReverseGeocodeUSLocation Output Data Options

optionName	Description
OutputRecordType	Specifies the optional data to include in the output. Note that ReverseGeocodeUSLocation always returns the data listed in **Default Output** on page 316. The data you select here is returned with the default output data. • **C**—Census • **L**—Latitude/Longitude • **E**—Parsed Elements • **Q**—Qualifiers • **R**—Range • **S**—Segment For a list of the fields included in each data type, see **Output** on page 314. If you do not want all of the fields in a record type returned, do not specify the record type. Instead, list only those fields you want returned in OutputFields.
OutputFields	Specifies the individual output fields you want returned. List fields with a pipe (|) between each field. You can use this field instead of the Output Record Type to limit the output to those fields that are important to your data needs. Default list: AddressLine1|LastLine|Longitude|Latitude|MatchCode|LocationCode

Output

ReverseGeocodeUSLocation returns the following categories of data:

- **Census** on page 315
- **Default Output** on page 316
- **Latitude/Longitude** on page 318
- **Parsed Elements** on page 318

- **Qualifiers** on page 319
- **Range** on page 322
- **Segment** on page 322

Census

The Census output record type contains census information from the U.S. 2000 Census. To include census data in the output, set **OutputRecordType** = C.

Table 89: Census Output Fields

columnName	Max. Field Length (bytes)	Description
APN	46	The assessor's parcel number of the property. The assessor's parcel number is an ID number assigned to a property by the local property tax authority.
BlockSuffix	2	Single character block suffix for split Census blocks. Returns A or B. Only available in Centrus Enhanced data.
CBSACode	6	Indicates Core Based Statistical Area (CBSA).
CBSADivisionCode	6	Denotes a subdivision of a CBSA.
CBSADivisionName	128	Describes a subdivision of a CBSA. Only returned when you set the configuration options to return additional descriptions (verbose).
CBSAMetro	2	Metropolitan Statistical Area. Valid values include: **Y** Metro statistical area. **N** Micro statistical area. **null** Data unavailable.
CBSAName	128	Describes CBSA. Only returned when you set the configuration options to return additional descriptions (verbose).
CensusBlockID	16	The ID of the Census Federal Information Processing Standard (FIPS) code.
CensusTract	7	6-digits extracted from the CensusBlockID.
CSACode	4	Denotes the code for a geographic entity that consists of 2 or more adjacent CBSAs with employment interchange measures of at least 15.
CSAName	128	Describes the name for a geographic entity that consists of 2 or more adjacent CBSAs with employment interchange measures of at least 15. Only returned when you set the configuration options to return additional descriptions (verbose).
USCountyName	128	Name of the county, including the text "County" or "Parish." Only returned when you set the configuration options to return additional descriptions (verbose).

columnName	Max. Field Length (bytes)	Description
USFIPSCountyNumber	4	3-digit FIPS county code extracted from the CensusBlockID.
USFIPSStateCode	3	2-digit FIPS state code extracted from the CensusBlockID.
USFIPSStateCountyCode	6	5-digit FIPS code for state and county extracted from the CensusBlockID.

Default Output

ReverseGeocodeUSAddress always includes the following fields in the output.

Table 90: Default Output Fields

columnName	Max. Field Length (bytes)	Description
AdditionalInputData	61	This field is populated with information from the mail stop output field. **Note:** ReverseGeocodeUSLocation does not process this information. It simply includes the information as entered in the input data.
AddressLine1	104	First line of the address.
AddressLine2	104	Second line of the address.
City	29	Municipality name.
Confidence	4	Indicates the confidence in the output provided. The range is from 0 (zero) to 100, with 0 being no match and 100 being an exact match.
Country	25	The name of the country. Since ReverseAPNLookup only works for U.S. locations, this field will always contain **United States of America**.
Distance	8	The distance, in feet, from the input geocode to nearest matched address or intersection.
Elevation	11	The location's elevation in feet above or below sea level.
FirmName	41	Name of the company.
LastLine	61	Complete last address line (municipality, state, and postal code).
Latitude	11	7-digit number in degrees and calculated to 4 decimal places (in the format specified).
LocationCode	5	Indicate the accuracy (quality) of the assigned geocode. For more information, see **Address Location Codes** on page 328.

columnName	Max. Field Length (bytes)	Description
Longitude	12	7-digit number in degrees and calculated to 4 decimal places (in the format specified).
MatchCode	5	Indicates the portions of the address that matched to the directory file. For more information, see **Geocoding Match Codes** on page 341.
PercentGeocode	6	The percent along the street segment that matches the geocode. For example, if the returned geocode falls 1/3 along the way of the entire street segment, the percent is 33.000. **Note:** This value is always 0.0 for matches to point-level data and intersections.
PostalCode	10	9-digit ZIP Code with or without a hyphen.
PostalCode.AddOn	5	4-digit ZIP Code extension.
PostalCode.Base	6	5-digit ZIP Code.
ProcessedBy	4	The underlying software that processed the request. KGR for ReverseGeocodeUSLocation.
RRHC	2	Rural Route Highway Contract (RRHC). This field is null if the address not a RRHC.
StateProvince	3	2-character state abbreviation.
Status	2	Reports the success or failure of the match attempt **null** Success **F** Failure
Status.Code	23	Reason for failure: • Internal System Error • No Geocode Found • Insufficient Input Data
Status.Description	32	Description of the problem: **Problem + explanation** Returned when Status.Code contains "Internal System Error." **Geocoding Failed** Returned when Status.code contains "No Geocode Found". **No location returned** Returned when Status.code contains "No Geocode Found."
StreetDataType	20	The data set that ReverseGeocodeUSLocation attempted to match against. **USPS** USPS **TIGER** TIGER

columnName	Max. Field Length (bytes)	Description
		TOMTOM TomTom Streets data set **SANBORN POINT DATA** Sanborn Points data set **NAVTEQ** NAVTEQ Streets data set **TOMTOM POINT DATA** TomTom Points data set **AUXILIARY** Auxiliary file **CENTRUS POINT DATA** Centrus Points data set **USER DICTIONARY** User Dictionary **NAVTEQ POINT DATA** NAVTEQ Points data set **MASTER LOCATION** Master Location Data
StreetSide	2	Indicates the side of the street the range occupies. One of the following: **L** The range occupies the left side of the street. **R** The range occupies the right side of the street. **B** The range occupies both sides of the street. **U** Undetermined.
USUrbanName	31	Urbanization name. Puerto Rico addresses only.

Latitude/Longitude

The latitude/longitude output fields contain the geographic coordinates of the location. To include latitude/longitude output fields in the output, set **OutputRecordType** = L.

Table 91: Latitude/Longitude Output Fields

columnName	Max. Field Length (bytes)	Description
Elevation	11	The location's elevation in feet above or below sea level.
Latitude	11	7-digit number in degrees and calculated to 4 decimal places (in the format specified).
Longitude	12	7-digit number in degrees and calculated to 4 decimal places (in the format specified).

Parsed Elements

The Parsed Elements output record type contains standard address information as individual units, such as street suffixes (AVE) and leading directionals (N and SE). To include parsed elements in the output, set **OutputRecordType** = E.

Table 92: Parsed Elements Output Fields

columnName	Max. Field Length (bytes)	Description
ApartmentLabel	5	Unit, such as apartment, suite, or lot.
ApartmentLabel2	5	Unit, such as apartment, suite, or lot.
ApartmentNumber	12	Unit number.
ApartmentNumber2	12	Unit number.
CrossStreetLeadingDirectional	3	Prefix for cross street.
CrossStreetName	41	Name of cross street.
CrossStreetSuffix	5	Cross street suffix.
CrossStreetTrailingDirectional	3	Postfix for cross street.
HouseNumber	12	Building number for the matched location. **Note:** This is an approximate building number based on the latitude and longitude provided. This approximate address may not exist or may not accept mail delivery.
LeadingDirectional	3	Street directional that precedes the street name. For example, the N in 138 N Main Street.
PrivateMailbox	9	Private mailbox. Not output for multiline input.
PrivateMailbox.Designator	5	Private mailbox description. Not output for multiline input.
StreetName	41	Street name.
StreetSuffix	5	The street type of the matched location. For example, AVE for Avenue.
TrailingDirectional	3	Street directional that follows the street name. For example, the N in 456 Washington N.

Qualifiers

The qualifiers output record type contains qualification information on the match, such as the location code and the match code. To include latitude/longitude output fields in the output, set **OutputRecordType** = Q.

Table 93: Qualifiers Output Fields

columnName	Max. Field Length (bytes)	Description
CountryLevel	2	The category of postal data available. Always returns A in ReverseGeocodeUSLocation—Validates, corrects, and

columnName	Max. Field Length (bytes)	Description
		provides missing postal code, city name, state/county name, street address elements, and country name.
DatabaseVersion	15	USPS publish date, in the format Month Year.
EWSMatch	2	Indicates if ReverseGeocodeUSLocation denied a match because of Early Warning System (EWS) data. **Y** EWS denied a match. **null** EWS did not deny a match. For more information on EWS, see **Early Warning System (EWS)** on page 212.
ExpirationDate	11	Date the database expires, in the format MM/DD/YYYY.
GeoStanMatchScore	13	Record matching score (for multimatches only).
Intersection	2	Indicates if ReverseGeocodeUSLocation found a cross-street match. **T** True, a cross-street match was found. **F** False, a cross-street match was not found.
IsAlias	4	ReverseGeocodeUSLocation located a matched record by an index alias. Returns 3 characters. The first is an **N** for normal street match or **A** for alias match (including buildings, aliases, firms, etc.). The next 2 characters are: **01** Basic index (normal address match) **02** USPS street name alias index **03** USPS building index **04** USPS firm name index **05** Statewide intersection alias match (when using the Usw.gsi or Use.gsi file) **06** Spatial data street name alias (when using, the Us_pw.gsi, Us_pe.gsi, Us_psw.gsi, or Us_pse.gsi file is required) **07** Alternate index (when using Zip9.gsu, Zip9e.gsu, and Zip9w.gsu) **08** LACS^{Link} **09** Auxiliary file match **10** Centrus Alias index (when using usca.gsi)
LACSAddress	2	Indicates if ReverseGeocodeUSLocation converted an address due to the Locatable Address Conversion System (LACS). **L** Converted **null** Not converted.

columnName	Max. Field Length (bytes)	Description
		For more information on LACS, see **Locatable Address Conversion System (LACS)** on page 211.
LocationCode.Description	128	LocationCode converted to text. Only returned when you set the configuration options to return additional descriptions (verbose).
MatchCode.Description	128	MatchCode converted to text. Only returned when you set the configuration options to return additional descriptions (verbose).
RecordType	18	Indicates the record type: • GeneralDelivery • HighRise • FirmRecord • Normal • PostOfficeBox • RRHighwayContract
RecordType.Default	2	Indicates type of match that occurred for the record type HighRise or RRHighwayContract: **Y** Default match. **N** Exact match. **U** Not matched.
StreetDataCode	3	Indicates the data used to obtain a match. **0** USPS data in either the Centrus Enhanced, Centrus TomTom, or Centrus NAVTEQ database. **1** TIGER data in the Centrus Enhanced database. **2** TomTom data in the Centrus TomTom database. **6** NAVTEQ data in the Centrus NAVTEQ database. **7** TomTom point-level data in the Centrus TomTom Points database. **8** Point-level data from the Centrus Points database. **9** Auxiliary file data. **10** User Dictionary. **11** Point-level data from NAVTEQ Points database. **12** Master Location Data (MLD). For more information about these databases, see **Enterprise Geocoding Databases** on page 205.
StreetDataType	20	Indicates the data first used to attempt a match.

Range

The range output record type contains information on the street range, such as the high and low unit numbers. To include range data fields in the output, set **OutputRecordType** = R.

Table 94: Range Output Fields

columnName	Max. Field Length (bytes)	Description
Alternate	2	USPS code that specifies whether a record is a base or alternate record. **B** Base record. Base records can represent a range of addresses or an individual address, such as a firm record. **A** Alternate record. Alternate records are individual delivery points.
HouseNumberHigh	12	House number high.
HouseNumberLow	12	House number low.
HouseNumberParity	2	Indicates if the house number range contains even or odd numbers. **E** Even **O** Odd **B** Both
PostalCodeExtensionHigh	5	4-digit ZIP Code extension high.
PostalCodeExtensionLow	5	4-digit Zip Code extension low.
UnitNumberHigh	12	Unit number high.
UnitNumberLow	12	Unit number low.
UnitNumberParity	2	Indicates if the unit number range contains even or odd numbers. **E** Even **O** Odd **B** Both

Segment

The segment output record type contains information on the street segment identified by the data provider. To include segment data fields in the output, set **OutputRecordType** = S.

Table 95: Segment Output Fields

columnName	Max. Field Length (bytes)	Description
BlockLeft	16	Provides the Census FIPS Code that indicates the address is on the left side of the street.
BlockRight	16	Provides the Census FIPS Code that indicates the address is on the right side of the street.
BlockSuffixLeft	2	Current left Block suffix for Census 2000 Geography. Returns A or B. Only available in Centrus Enhanced data.
BlockSuffixRight	2	Current right Block suffix for Census 2000 Geography. Returns A or B. Only available in Centrus Enhanced data.
RoadClass	3	The type of road: **1** Major **2** Minor
SegmentCode	11	Unique 10-digit segment ID assigned by the street network provider.
SegmentDirection	2	Indicates the order of numbers on a segment. **F** Forward **R** Reversed **B** Both **U** Undetermined
SegmentHouseNumberHigh	12	A high range number in the segment.
SegmentHouseNumberLow	12	A low range number in the segment.
SegmentLength		The length, in feet, of a block segment. *This field is deprecated.*
SegmentParity	2	Indicates which side of the street has odd numbers. **L** Left side of the street **R** Right side of the street **B** Both sides of the street **U** Undetermined

Geocode US Address Auxiliary Files

Auxiliary File Overview

Use auxiliary files to match against special data that is not included in the GeocodeUSAddress database.

The GeocodeUSAddress database is updated regularly to incorporate changes made by the USPS and third-party data vendors. You may have newer information that has not yet been incorporated. Auxiliary files provide a way for you to process your input records against a file that includes these changes.

Note: ReverseGeocodeUSAddress does not support auxiliary files.

There are two types of auxiliary file records:

- **Street Records**—Contains a range of one or more addresses on a street. For required fields, see **Auxiliary File Layout** on page 325. A street record must not have secondary address information mailstops, Private mail boxes (PMBs), and PO Boxes.
- **Landmark Records**—Represents a single site. For required fields, see **Auxiliary File Layout** on page 325. A landmark record must not have street type abbreviations, predirectional and postdirectional abbreviations, or low and high house numbers.

Note: You cannot update the auxiliary file while GeocodeUSAddress is running. If you want to update the auxiliary file, stop GeocodeUSAddress before attempting to replace or edit the file.

Matching to Auxiliary Files

GeocodeUSAddress matches an input address to an auxiliary file as follows:

1. GeocodeUSAddress determines if there is an auxiliary file present.

   If you have an auxiliary file in the dataset directory, GeocodeUSAddress automatically loads and attempts to match to the auxiliary file. You can verify that GeocodeUSAddress found an auxiliary file by looking at the version information page in the Management Console. One of the following statuses display:

   - Loaded—An auxiliary file is loaded
   - None—An auxiliary file has not been found or loaded
   - Invalid—An auxiliary file was found, but failed to successfully load

   GeocodeUSAddress only accepts one auxiliary file. If more than one auxiliary files is present, GeocodeUSAddress attempts to match against the first file. GeocodeUSAddress ignores any additional auxiliary files for matching, regardless if GeocodeUSAddress found a match to the first auxiliary file.

   If a record in the auxiliary files is invalid, GeocodeUSAddress returns a invalid record message. GeocodeUSAddress continues to match input addresses with the auxiliary file, but will not match to the invalid auxiliary file record.

2. If an auxiliary file is present, GeocodeUSAddress attempts to match to the auxiliary file.

   GeocodeUSAddress assumes that the auxiliary file is the most accurate data set and attempts to find a match to the input address in the auxiliary file. If GeocodeUSAddress cannot find a match in the auxiliary file, it matches the input address with the other Enterprise Geocoding Module databases.

   Note: GeocodeUSAddress only matches input address lists to auxiliary files if there is an exact match. Your input address list should be free of misspellings and incomplete addresses.

3. If GeocodeUSAddress finds an exact record match to the auxiliary file, it standardizes the match to USPS regulations and returns the output of the auxiliary file match.

   GeocodeUSAddress uses the following defaults if you do not include the values in the auxiliary file:

   - House number parity = B (both odds and evens)
   - Segment direction = A (ascending)
   - Side of street = U (unknown)

Record Type Matching Rules

When matching against an auxiliary file, GeocodeUSAddress uses the following rules:

Street record match

- The input house number must fall within or be equal to the low and high house number values of the auxiliary record.

- The input house number must agree with the parity of the auxiliary record.
- The input ZIP Code must exactly match the ZIP Code of the auxiliary record.

Landmark record match

- The input data must contain a ZIP Code and address line, and the values must exactly match the values on the auxiliary record.
- The input address cannot have any other data, such as a house number, unit number, or Private Mail Box (PMB).

Note: GeocodeUSAddress only matches the ZIP Code against the auxiliary file. GeocodeUSAddress does not verify that the ZIP Code of the input address record is correct for the city and state. Validate this information in your input lists before processing against the auxiliary file.

Unavailable Features and Functions

The following features and functions do not apply when GeocodeUSAddress makes an auxiliary file match.

- GeocodeUSAddress does not match to
- two-line addresses
- multi-line addresses
- intersection addresses
- dual addresses
- GeocodeUSAddress does not perform EWS, ZIPMove, LACSLink, or DPV processing on auxiliary matches
- You can only access the auxiliary file with processing through the Find function. You cannot access the auxiliary file through the Find First/Next or MBR functions
- You can only access the auxiliary file logic using the address code option of the Find function, not the geocode option.

Auxiliary Match Output

GeocodeUSAddress provides special data type, match codes, and location code values for auxiliary matches. When GeocodeUSAddress finds a match to an auxiliary file, the default output follows these conventions:

- GeocodeUSAddress formats the auxiliary file match as a street-style address for output. This excludes PO Boxes, Rural Routes, General Delivery, etc.
- GeocodeUSAddress follows the case setting you indicate (by default, upper case) by the casing function. GeocodeUSAddress does not maintain the casing in the auxiliary file for mixed casing values. For example, GeocodeUSAddress returns O'Donnell as ODONNELL or Odonnell depending on the setting of the casing function.

Note: GeocodeUSAddress does not change the casing for the User Data field.

- GeocodeUSAddress removes spaces at the beginning and ending of fields in the auxiliary file.

Note: GeocodeUSAddress does not remove spaces for the User Data field.

Auxiliary File Layout

You must comply with the following organizational rules when you create auxiliary file:

- Files are fixed-width text files with a `.gax` extension
- Files can contain up to 500,000 records.
- Use semicolons in the first column to indicate a row is a comment, not a data record; GeocodeUSAddress ignores rows that begin with a semicolon.
- For optimal performance, order the records within the file by descending ZIP Code, and then descending street name.

- Records must represent only one side of a street. To represent both sides of a street, create a record for each side of the street.
- Records must represent segments that are straight lines.
- House numbers must follow USPS rules documented in Publication 28.
- Numeric fields, such as ZIP Codes, must contain only numbers.
- If house numbers are present in the record, the house number range must be valid according to USPS rules documented in Publication 28, Appendix E.
- Latitude and Longitude values must be in millionths of decimal degrees.
- Records cannot contain PO Box addresses.

The following table shows auxiliary file layout.

Table 96: Auxiliary File Layout

Field	Description	Required	Required for Street Segment Match	Required for Landmark Match	Exact Match Required if Present	Length	Position
ZIP Code	5-digit ZIP Code.	X	X	X	X	5	1-5
Street name	Name of the street or landmark.	X	X	X	X	30	6-35
Street type abbreviation	Street type. Also called street suffix. See the USPS Publication 28, Appendix C for a complete list of supported street types.				X	4	36-39
Predirectional	USPS street name predirectional abbreviation. Supported values are N, E, S, W, NE, NW, SE, and SW.				X	2	40-41
Postdirectional	USPS street name postdirectional abbreviations. Supported values are N, E, S, W, NE, NW, SE, and SW.				X	2	42-43
RESERVED	RESERVED					4	44-47
Low house number	Low house number of the address range.	X	X			11	48-58
High house number	High house number of the address range.	X	X			11	59-69
House number parity	Indicates the parity of the house number in the range.					1	70

Field	Description	Required	Required for Street Segment Match	Required for Landmark Match	Exact Match Required if Present	Length	Position
	E - Even O - Odd B - Both						
Segment direction	Direction the house numbers progress along the segment: F - Forward (default) R - Reverse					1	71
RESERVED	RESERVED					1	72
FIPS state	US government FIPS state code.					2	73-74
FIPS county	US government FIPS county code.					3	75-77
Census tract	US Census tract number.					6	78-83
Census block group	US Census block group number.					1	84
Census block ID	US Census block ID number.					3	85-87
RESERVED	RESERVED					5	88-92
State abbreviation	USPS state abbreviation.					2	93-94
County name	Name of the county.					25	95-119
MCD code	Minor Civil Division code.					5	120-124
MCD name	Minor Civil Division name.					40	125-164
CBSA code	Core Based Statistical Area code.					5	165-169
CBSA name	Core Based Statistical Area name.					49	170-218
RESERVED	RESERVED					5	219-223
City Name	City name. Overrides the city/state preferred city name upon a return.					40	224-263

Field	Description	Required	Required for Street Segment Match	Required for Landmark Match	Exact Match Required if Present	Length	Position
RESERVED	RESERVED					237	264-500
User-defined data	User-defined data.					300	501-800
Record ID Number	User-defined unique record identifier.					10	801-810
Side of street	Side of the street for the address: L - Left side R - Right side B - Both sides U - Unknown side (default) This is relative to the segment endpoints and the segment direction.					1	811
Beginning longitude	Beginning longitude of the street segment in millionths of degrees.	X	X	X		11	812-822
Beginning latitude	Beginning latitude of the street segment in millionths of degrees.	X	X	X		10	823-832
Ending longitude	Ending longitude of the street segment in millionths of degrees.					11	833-843
Ending latitude	Ending latitude of the street segment in millionths of degrees.					10	844-853

Location and Match Codes for U.S. Geocoding

Address Location Codes

Location codes that begin with an "A" are address location codes. Address location codes indicate a geocode made directly to a street network segment (or two segments, in the case of an intersection).

An address location code has the following characters.

1st character	Always an A indicating an address location.
2nd character	May be one of the following

	C	Interpolated address point location
	G	Auxiliary file data location
	I	Application infers the correct segment from the candidate records
	P	Point-level data location
	R	Location represents a ranged address
	S	Location on a street range
	X	Location on an intersection of two streets
3rd and 4th character	Digit indicating other qualities about the location.	

Table 97: Address Location Codes

Code		Description
AGn		Indicates an auxiliary file for a geocode match where n is one of the following values:
	n = 0	The geocode represents the center of a parcel or building.
	n = 1	The geocode is an interpolated address along a segment.
	n = 2	The geocode is an interpolated address along a segment, and the side of the street cannot be determined from the data provided in the auxiliary file record.
	n = 3	The geocode is the midpoint of the street segment.
APnn		Indicates a point-level geocode match representing the center of a parcel or building, where nn is one of the following values:
	nn = 02	Parcel centroid Indicates the center of an accessor's parcel (tract or lot) polygon. When the center of an irregularly shaped parcel falls outside of its polygon, the centroid is manually repositioned to fall inside the polygon as closely as possible to the actual center.
	nn = 04	Address points Represents field-collected GPS points with field-collected address data.
	nn = 05	Structure centroid

Code	Description
	Indicates the center of a building footprint polygon, where the building receives mail or has telephone service. Usually a residential address consists of a single building. For houses with outbuildings (detached garages, shed, barns, etc.), only the residences have a structure point. Condominiums and duplexes have multiple points for each building. Larger buildings, such as apartment complexes, typically receive mail at one address for each building and therefore individual apartments are not represented as discrete structure points. Shopping malls, industrial complexes, and academic or medical center campuses where one building accepts mail for the entire complex are represented as one point. When addresses are assigned to multiple buildings within one complex, each addressed structure is represented by a point. If the center of a structure falls outside of its polygon, the center is manually repositioned to fall inside the polygon.
`nn = 07`	Manually placed Address points are manually placed to coincide with the midpoint of a parcel's street frontage at a distance from the center line.
`nn = 08`	Front door point Represents the designated primary entrance to a building. If a building has multiple entrances and there is no designated primary entrance or the primary entrance cannot readily be determined, the primary entrance is chosen based on proximity to the main access street and availability of parking.
`nn = 09`	Driveway offset point Represents a point located on the primary access road (most commonly a driveway) at a perpendicular distance of between 33-98 feet (10-30 meters) from the main roadway.
`nn = 10`	Street access point Represents the primary point of access from the street network. This address point type is located where the driveway or other access road intersects the main roadway.
`nn = 21`	Base parcel point

Code		Description
		When unable to match to an input unit number, or when the unit number is missing from an address location with multiple units, the "base" parcel information is returned, the address is not standardized to a unit number, and additional information, such as an Assessor's Parcel Number, is not returned.
	`nn = 22`	Backfill address point The precise parcel centroid is unknown. The address location assigned is based on two known parcel centroids.
	`nn = 23`	Virtual address point The precise parcel centroid is unknown. The address location assigned is relative to a known parcel centroid and a street segment end point.
	`nn = 24`	Interpolated address point The precise parcel centroid is unknown. The address location assigned is based on street segment end points.
`AIn`		The correct segment is inferred from the candidate records at match time.
`ASn`		House range address geocode. This is the most accurate geocode available.
AIn and ASn share the same qualities for n as follows:		
	`n = 0`	Best location.
	`n = 1`	Street side is unknown. The Census FIPS Block ID is assigned from the left side; however, there is no assigned offset and the point is placed directly on the street.
	`n = 2`	Indicates one or both of the following: • The address is interpolated onto a TIGER segment that did not initially contain address ranges. • The original segment name changed to match the USPS spelling. This specifically refers to street type, predirectional, and postdirectional. **Note:** Only the second case is valid for non-TIGER data because segment range interpolation is only completed for TIGER data.
	`n = 3`	Both 1 and 2.

Code		Description
	n = 7	Placeholder. Used when starting and ending points of segments contain the same value and shape data is not available.
ACnh		Indicates a point-level geocode that is interpolated between 2 parcel centroids (points), a parcel centroid and a street segment endpoint, or 2 street segment endpoints.
	The ACnn 4th digit characteristics are as follows:	
	n = 0	Represents the interpolation between two points, both coming from User Dictionaries.
	n = 1	Represents the interpolation between two points. The low boundary came from a User Dictionary and the high boundary, from a non-User Dictionary.
	n = 2	Represents the interpolation between one point and one street segment end point, both coming from User Dictionaries.
	n = 3	Represents the interpolation between one point (low boundary) and one street segment end point (high boundary). The low boundary came from a User Dictionary and the high boundary from a non-User Dictionary.
	n = 4	Represents the interpolation between two points. The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.
	n = 5	Represents the interpolation between two points, both coming from non-User Dictionaries.
	n = 6	Represents the interpolation between one point (low boundary) and one street segment end point (high boundary). The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.
	n = 7	Represents the interpolation between one point and one street segment end point and both came from non-User Dictionaries.
	n = 8	Represents the interpolation between one street segment end point andone point, both coming from User Dictionaries.
	n = 9	Represents the interpolation between one street segment end point (low boundary) andone point (high boundary). The low boundary came from a User Dictionary and

Code		Description
		the high boundary from a non-User Dictionary.
	n = A	Represents the interpolation between two street segment end points, both coming from User Dictionaries.
	n = B	Represents the interpolation between two street segment end points. The low boundary came from a User Dictionary and the high boundary from a non-User Dictionary.
	n = C	Represents the interpolation between one street segment end point (low boundary) and one point (high boundary). The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.
	n = D	Represents the interpolation between one street segment end point and one point, both coming from non-User Dictionary.
	n = E	Represents the interpolation between two street segment end points. The low boundary came from a non-User Dictionary and the high boundary from a User Dictionary.
	n = F	Represents the interpolation between two street segment end points, both coming from non-User Dictionaries.
ARn		Ranged address geocode, where n is one of the following:
	n = 1	The geocode is placed along a single street segment, midway between the interpolated location of the first and second input house numbers in the range.
	n = 2	The geocode is placed along a single street segment, midway between the interpolated location of the first and second input house numbers in the range, and the side of the street is unknown. The Census FIPS Block ID is assigned from the left side; however, there is no assigned offset and the point is placed directly on the street.
	n = 4	The input range spans multiple USPS segments. The geocode is placed on the endpoint of the segment which corresponds to the first input house number, closest to the end nearest the second input house number.
	n = 7	Placeholder. Used when the starting and ending points of the matched segment

Code		Description
		contain the same value and shape data is not available.
AXn		Intersection geocode, where n is one of the following:
	n = 3	Standard single-point intersection computed from the center lines of street segments.
	n = 8	Interpolated (divided-road) intersection geocode. Attempts to return a centroid for the intersection.

Street Centroid Location Codes

Location codes that begin with "C" are street centroid location codes. Street centroid location codes indicate the Census ID accuracy and the position of the geocode on the returned street segment. Street centroids may be returned if the street centroid fallback option is enabled and an address-level geocode could not be determined.

A street centroid location code has the following characters.

1st character	Always C indicating a location derived from a street segment.
2nd character	Census ID accuracy based on the search area used to obtain matching Street Segment.
3rd character	Location of geocode on the returned street segment.

The following table contains the values and descriptions for the location codes.

Character position	Code	Description
2nd Character		
	B	Block Group accuracy (most accurate). Based on input ZIP Code.
	T	Census Tract accuracy. Based on input ZIP Code.
	C	Unclassified Census accuracy. Normally accurate to at least the County level. Based on input ZIP Code.
	F	Unknown Census accuracy. Based on Finance area.
	P	Unknown Census accuracy. Based on input City.
3rd Character		

Character position	Code	Description
	C	Segment Centroid.
	L	Segment low-range end point.
	H	Segment high-range end point.

ZIP + 4 Centroid Location Codes

Location codes that begin with a "Z" are ZIP + 4 centroid location codes. ZIP + 4 centroids indicate a geocode could not be determined for the address, so the location of the center of the address's ZIP + 4 was returned instead. ZIP + 4 centroid location codes indicate the quality of two location attributes: Census ID accuracy and positional accuracy.

A ZIP + 4 centroid location code has the following characters.

1st character	Always Z indicating a location derived from a ZIP centroid.
2nd character	Census ID accuracy.
3rd character	Location type.
4th character	How the location and Census ID was defined. Provided for completeness, but may not be useful for most applications.

Table 98: ZIP + 4 Centroid Location Codes

Character Position	Code	Description
2nd Character		
	B	Block Group accuracy (most accurate).
	T	Census Tract accuracy.
	C	Unclassified Census accuracy. Normally accurate to at least the County level.
3rd Character		
	5	Location of the Post Office that delivers mail to the address, a 5-digit ZIP Code centroid, or a location based upon locale (city). See the 4th character for a precise indication of locational accuracy.
	7	Location based upon a ZIP + 2 centroid. These locations can represent a multiple block area in

Character Position	Code	Description
		urban locations, or a slightly larger area in rural settings.
	9	Location based upon a ZIP + 4 centroid. These are the most accurate centroids and normally place the location on the correct block face. For a small number of records, the location may be the middle of the entire street on which the ZIP + 4 falls. See the 4th character for a precise indication of locational accuracy.
4th Character		
	A	Address matched to a single segment. Location assigned in the middle of the matched street segment, offset to the proper side of the street.
	a	Address matched to a single segment, but the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	B	Address matched to multiple segments, all segments have the same Block Group. Location assigned to the middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	b	Same as methodology B except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	C	Address matched to multiple segments, with all segments having the same Census Tract. Returns the Block Group representing the most households in this ZIP + 4. Location assigned to t he middle of the matched street segment with the most house number ranges within this

Character Position	Code	Description
		ZIP + 4. Location offset to the proper side of the street.
	c	Same as methodology C except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	D	Address matched to multiple segments, with all segments having the same County. Returns the Block Group representing the most households in this ZIP + 4. Location assigned to the middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	d	Same as methodology D except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	E	Street name matched; no house ranges available. All matched segments have the same Block Group. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.
	F	Street name matched; no house ranges available. All matched segments have the same Census Tract. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.
	G	Street name matched (no house ranges available). All matched segments have the same County. Location placed on the segment closest to the center of the matched segments. In most

Character Position	Code	Description
		cases, this is on the mid-point of the entire street.
	H	Same as methodology G, but some segments are not in the same County. Used for less than .05% of the centroids.
	I	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, and b. All centroids in this ZIP + 2 cluster have the same Block Group. Location assigned to the ZIP + 2 centroid.
	J	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, b, C, and c. All centroids in this ZIP + 2 cluster have the same Census Tract. Location assigned to the ZIP + 2 centroid.
	K	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, b, C, c, D, and d. Location assigned to the ZIP + 2 centroid.
	L	Created ZIP + 2 cluster centroid as defined by methodology E. All centroids in this ZIP + 2 cluster have the same Block Group. Location assigned to the ZIP + 2 centroid.
	M	Created ZIP+2 cluster centroid as defined by methodology E and F. All centroids in this ZIP + 2 cluster have the same Census Tract. Location assigned to the ZIP + 2 centroid.
	N	Created ZIP + 2 cluster centroid as defined by methodology E, F, G, and H. Location assigned to the ZIP + 2 centroid.
	O	ZIP Code is obsolete and not currently used by the USPS. Historic location assigned.
	V	Over 95% of addresses in this ZIP Code are in a single Census Tract. Location assigned to the ZIP Code centroid.
	W	Over 80% of addresses in this ZIP Code are in a single Census

Character Position	Code	Description
		Tract. Reasonable Census Tract accuracy. Location assigned to the ZIP Code centroid.
	X	Less than 80% of addresses in this ZIP Code are in a single Census Tract. Census ID is uncertain. Location assigned to the ZIP Code centroid.
	Y	Rural or sparsely populated area. Census code is uncertain. Location based upon the USGS places file.
	Z	P.O. Box or General Delivery addresses. Census code is uncertain. Location based upon the Post Office location that delivers the mail to that address.

Geographic Centroid Location Codes

Location codes that begin with "G" are geographic centroid location codes. Geographic centroids may be returned if the street centroid fallback option is enabled and an address-level geocode could not be determined. Geographic centroid location codes indicate the quality a city, county, or state centroid.

A geographic centroid location code has the following characters.

1st character	Always G indicating a location derived from a geographic centroid.
2nd character	Geographic area type. One of the following: **M** Municipality (for example, a city) **C** County **S** State

Address Unavailable

Location codes that begin with "E" indicate that neither an address location nor a ZIP + 4 centroid could be determined. This usually occurs when you have requested ZIP Code centroids of a high quality, and one is not available for that match.

An unavailable address code has the following characters.

Table 99: Match Codes for No Match

Code	Description
Ennn	Indicates an error, or no match. This can occur when the address entered does not exist in the database, or the address is badly

Code	Description
	formed and cannot be parsed correctly. The last three digits of an error code indicate which parts of an address the application could not match to the database.
nnn = 000	No match made.
nnn = 001	Low level error.
nnn = 002	Could not find data file.
nnn = 003	Incorrect GSD file signature or version ID.
nnn = 004	GSD file out of date. Only occurs in CASS mode.
nnn = 010	No city and state or ZIP Code found.
nnn = 011	Input ZIP not in the directory.
nnn = 012	Input city not in the directory.
nnn = 013	Input city not unique in the directory.
nnn = 014	Out of licensed area. Only occurs if using Pitney Bowes Software licensing technology.
nnn = 015	Record count is depleted and license has expired.
nnn = 020	No matching streets found in directory.
nnn = 021	No matching cross streets for an intersection match.
nnn = 022	No matching segments.
nnn = 023	Unresolved match.
nnn = 024	No matching segments. (Same as 022.)
nnn = 025	Too many possible cross streets for intersection matching.
nnn = 026	No address found when attempting a multiline match.
nnn = 027	Invalid directional attempted.
nnn = 028	Record also matched EWS data, therefore the application denied the match.

Code	Description
nnn = 029	No matching range, single street segment found.
nnn = 030	No matching range, multiple street segments found.

Geocoding Match Codes

Geocoding components return match codes indicating the address portions that matched or did not match to the database. If the geocoder cannot make a match, the match code begins with E and the remaining digits indicate why the address did not match. The digits do not specifically refer to which address elements did not match, but rather why the address did not match.

Table 100: Match Codes

Code	Description
Ahh	Same as Shh, but indicates match to an alias name record or an alternate record.
Chh	The street address did not match, but the geocoder located a street segment based on the input ZIP Code or city .
D00	Matched to a small town with P.O. Box or General Delivery only.
Gxx	Matched to an auxiliary file.
Hhh	The house number was changed.
Jhh	Matched to a user-defined dictionary.
Nxx	Matched to the nearest address. Used with reverse geocoding. The following are the only values for N: **NSO** Nearest street center match (nearest street segment interpolated) **NS1** Nearest unranged street segment **NP0** Nearest point address **NX0** Nearest intersection
P	Successful reverse APN lookup.
Qhh	Matched to USPS range records with unique ZIP Codes. CASS rules prohibit altering an input ZIP if it matches a unique ZIP Code value.
Rhh	Matched to a ranged address.
Shh	Matched to USPS data. This is considered the best address match, because it matched directly against

Code	Description
	the USPS list of addresses. S is returned for a small number of addresses when the matched address has a blank ZIP + 4.
Thh	Matched to a street segment record. Street segment records do not contain ZIP Code information. If you enter a ZIP Code, the application returns the ZIP Code you entered. If the input city and state has only one ZIP Code, the application returns that ZIP Code.
Uhh	Matched to USPS data but cannot resolve the ZIP + 4 code without the firm name or other information. CASS mode returns an E023 (multiple match) error code.
Xhhh	Matched to an intersection of two streets, for example, "Clay St & Michigan Ave." The first hex digit refers to the last line information, the second hex digit refers to the first street in the intersection, and the third hex digit refers to the second street in the intersection. **Note:** The USPS does not allow intersections as a valid deliverable address.
Yhhh	Same as Xhhh, but an alias name record was used for one or both streets.
Z	No address given, but verified the provided ZIP Code .

The following table contains the description of the hex digits for the match code values.

Table 101: Description of Hex Digits

Code	In first hex position means:	In second and third hex position means:
0	No change in last line.	No change in address line.
1	ZIP Code changed.	Street type changed.
2	City changed.	Predirectional changed.
3	City and ZIP Code changed.	Street type and predirectional changed.
4	State changed.	Postdirectional changed.
5	State and ZIP Code changed.	Street type and postdirectional changed.
6	State and City changed.	Predirectional and postdirectional changed.

Code	In first hex position means:	In second and third hex position means:
7	State, City, and ZIP Code changed.	Street type, predirectional, and postdirectional changed.
8	ZIP + 4 changed.	Street name changed.
9	ZIP and ZIP + 4 changed.	Street name and street type changed.
A	City and ZIP + 4 changed.	Street name and predirectional changed.
B	City, ZIP, and ZIP + 4 changed.	Street name, street type, and predirectional changed.
C	State and ZIP + 4 changed.	Street name and postdirectional changed.
D	State, ZIP, and ZIP + 4 changed.	Street name, street type, and postdirectional changed.
E	State, City, and ZIP + 4 changed.	Street name, predirectional, and postdirectional changed.
F	State, City, ZIP, and ZIP + 4 changed.	Street name, street type, predirectional, and postdirectional changed.

If neither an address location nor a ZIP + 4 centroid can be determined, the location code will start with "E". This occurs infrequently when the component does not have a 5-digit centroid location. Enterprise Geocoding Module components can also return an E location code type when it cannot standardize an input address and there is no input ZIP Code. In this case, do not assume the ZIP Code returned with the non-standardized address is the correct ZIP Code because the component did not standardize the address; therefore, the component does not return geocoding or Census Block information.

Table 102: Match Codes for No Match

Code		Description
Ennn		Indicates an error, or no match. This can occur when the address entered does not exist in the database, or the address is badly formed and cannot be parsed correctly. The last three digits of an error code indicate which parts of an address the application could not match to the database.
	nnn = 000	No match made.
	nnn = 001	Low level error.
	nnn = 002	Could not find data file.

Code	Description
nnn = 003	Incorrect GSD file signature or version ID.
nnn = 004	GSD file out of date. Only occurs in CASS mode.
nnn = 010	No city and state or ZIP Code found.
nnn = 011	Input ZIP not in the directory.
nnn = 012	Input city not in the directory.
nnn = 013	Input city not unique in the directory.
nnn = 014	Out of licensed area. Only occurs if using Pitney Bowes Software licensing technology.
nnn = 015	Record count is depleted and license has expired.
nnn = 020	No matching streets found in directory.
nnn = 021	No matching cross streets for an intersection match.
nnn = 022	No matching segments.
nnn = 023	Unresolved match.
nnn = 024	No matching segments. (Same as 022.)
nnn = 025	Too many possible cross streets for intersection matching.
nnn = 026	No address found when attempting a multiline match.
nnn = 027	Invalid directional attempted.
nnn = 028	Record also matched EWS data, therefore the application denied the match.
nnn = 029	No matching range, single street segment found.
nnn = 030	No matching range, multiple street segments found.

Match Codes for U.S. Geocoding

Geocoding Match Codes

Geocoding components return match codes indicating the address portions that matched or did not match to the database. If the geocoder cannot make a match, the match code begins with E and the remaining digits indicate why the address did not match. The digits do not specifically refer to which address elements did not match, but rather why the address did not match.

Table 103: Match Codes

Code	Description
Ahh	Same as Shh, but indicates match to an alias name record or an alternate record.
Chh	The street address did not match, but the geocoder located a street segment based on the input ZIP Code or city .
D00	Matched to a small town with P.O. Box or General Delivery only.
Gxx	Matched to an auxiliary file.
Hhh	The house number was changed.
Jhh	Matched to a user-defined dictionary.
Nxx	Matched to the nearest address. Used with reverse geocoding. The following are the only values for N: **NSO** Nearest street center match (nearest street segment interpolated) **NS1** Nearest unranged street segment **NP0** Nearest point address **NX0** Nearest intersection
P	Successful reverse APN lookup.
Qhh	Matched to USPS range records with unique ZIP Codes. CASS rules prohibit altering an input ZIP if it matches a unique ZIP Code value.
Rhh	Matched to a ranged address.
Shh	Matched to USPS data. This is considered the best address match, because it matched directly against the USPS list of addresses. S is returned for a small number of addresses when the matched address has a blank ZIP + 4.
Thh	Matched to a street segment record. Street segment records do not contain ZIP Code

Code	Description
	information. If you enter a ZIP Code, the application returns the ZIP Code you entered. If the input city and state has only one ZIP Code, the application returns that ZIP Code.
Uhh	Matched to USPS data but cannot resolve the ZIP + 4 code without the firm name or other information. CASS mode returns an E023 (multiple match) error code.
Xhhh	Matched to an intersection of two streets, for example, "Clay St & Michigan Ave." The first hex digit refers to the last line information, the second hex digit refers to the first street in the intersection, and the third hex digit refers to the second street in the intersection. **Note:** The USPS does not allow intersections as a valid deliverable address.
Yhhh	Same as Xhhh, but an alias name record was used for one or both streets.
Z	No address given, but verified the provided ZIP Code .

The following table contains the description of the hex digits for the match code values.

Table 104: Description of Hex Digits

Code	In first hex position means:	In second and third hex position means:
0	No change in last line.	No change in address line.
1	ZIP Code changed.	Street type changed.
2	City changed.	Predirectional changed.
3	City and ZIP Code changed.	Street type and predirectional changed.
4	State changed.	Postdirectional changed.
5	State and ZIP Code changed.	Street type and postdirectional changed.
6	State and City changed.	Predirectional and postdirectional changed.
7	State, City, and ZIP Code changed.	Street type, predirectional, and postdirectional changed.
8	ZIP + 4 changed.	Street name changed.
9	ZIP and ZIP + 4 changed.	Street name and street type changed.

Code	In first hex position means:	In second and third hex position means:
A	City and ZIP + 4 changed.	Street name and predirectional changed.
B	City, ZIP, and ZIP + 4 changed.	Street name, street type, and predirectional changed.
C	State and ZIP + 4 changed.	Street name and postdirectional changed.
D	State, ZIP, and ZIP + 4 changed.	Street name, street type, and postdirectional changed.
E	State, City, and ZIP + 4 changed.	Street name, predirectional, and postdirectional changed.
F	State, City, ZIP, and ZIP + 4 changed.	Street name, street type, predirectional, and postdirectional changed.

If neither an address location nor a ZIP + 4 centroid can be determined, the location code will start with "E". This occurs infrequently when the component does not have a 5-digit centroid location. Enterprise Geocoding Module components can also return an E location code type when it cannot standardize an input address and there is no input ZIP Code. In this case, do not assume the ZIP Code returned with the non-standardized address is the correct ZIP Code because the component did not standardize the address; therefore, the component does not return geocoding or Census Block information.

Table 105: Match Codes for No Match

Code		Description
Ennn		Indicates an error, or no match. This can occur when the address entered does not exist in the database, or the address is badly formed and cannot be parsed correctly. The last three digits of an error code indicate which parts of an address the application could not match to the database.
	nnn = 000	No match made.
	nnn = 001	Low level error.
	nnn = 002	Could not find data file.
	nnn = 003	Incorrect GSD file signature or version ID.
	nnn = 004	GSD file out of date. Only occurs in CASS mode.
	nnn = 010	No city and state or ZIP Code found.

Code	Description
nnn = 011	Input ZIP not in the directory.
nnn = 012	Input city not in the directory.
nnn = 013	Input city not unique in the directory.
nnn = 014	Out of licensed area. Only occurs if using Pitney Bowes Software licensing technology.
nnn = 015	Record count is depleted and license has expired.
nnn = 020	No matching streets found in directory.
nnn = 021	No matching cross streets for an intersection match.
nnn = 022	No matching segments.
nnn = 023	Unresolved match.
nnn = 024	No matching segments. (Same as 022.)
nnn = 025	Too many possible cross streets for intersection matching.
nnn = 026	No address found when attempting a multiline match.
nnn = 027	Invalid directional attempted.
nnn = 028	Record also matched EWS data, therefore the application denied the match.
nnn = 029	No matching range, single street segment found.
nnn = 030	No matching range, multiple street segments found.

Result Codes for International Geocoding

Candidates returned by Spectrum geocoders return another class of return codes that are referred to as International Geocoding Result Codes. Each attempted match returns a result code in the Geocoder.MatchCode output field.

International Street Geocoding Result Codes (S Codes)

Street level geocoded candidates return a result code beginning with the letter S. The second character in the code indicates the positional accuracy of the resulting point for the geocoded record.

Table 106: Street (S) Result Codes

S Result Code	Description
S1	Single close match with the point located at postal code centroid.
S3	Single close match with the point located at postal code centroid.
S4	Single close match with the point located at the street centroid. The S4 code is followed by letters and dashes indicating match precision. see **Interpreting S Result Codes** on page 349
S5	Single close match with the point located at a street address position. The S5 code is followed by letters and dashes indicating match precision. For information about these letters, see **Interpreting S Result Codes** on page 349.
S7	Single match with the point located at an interpolated point along the candidate's street segment. When the potential candidate is not an address point candidate and there are no exact house number matches among other address point candidates, the S7 result is returned using address point interpolation. The point is interpolated according to the next highest or lowest address point candidate that both intersects the segment and whose house number is contained within the range of houses of the original candidate. By using known address reference points on the street segment, the S7 point can be adjusted to a more accurate position.
S8	Single close match with the point located at either the single point associated with an address point candidate or at an address point candidate that shares the same house number. No interpolation is required. S8 returns are possible with point databases only.
SX	Single close match with the point located at street intersection.

Interpreting S Result Codes

For S (street geocoded) international result codes, eight additional characters describe how closely the address matches an address in the database. The characters appear in the order listed in the following table. Any non-matched components are represented by a dash.

For example, the result code S5--N-SCZA represents a single close match that matched the street name, street suffix direction, town, and postcode. The dashes indicate that there was no match on house number, street prefix direction, or thoroughfare type. The match came from the Street Range Address database. This record would be geocoded at the street address position of the match candidate.

Category	Description	Example
H	House number	18
P	Street prefix direction P is present if any of these conditions are satisfied: • The candidate pre-directional matches the input pre-directional. • The candidate post-directional matches the input pre-directional after pre- and post-directionals are swapped. • The input does not have a pre-directional.	North
N	Street name	Merivale
T	Street type	St

Category	Description	Example
S	Street suffix direction S in result code is present if any of these conditions are satisfied: • The candidate post-directional matches the input post-directional. • The candidate pre-directional matches the input post-directional after pre- and post-directionals are swapped. • The input does not have a post-directional.	W
C	City name	South Brisbane
Z	Postal code	4101
A, G, or U	Database type used to obtain the match. • A—Street Range Address database. • U—Customer (user-defined) database.	A

International Postal Geocoding Result Codes (Z Codes)

Matches in the Z category indicate that a match was made at the postcode level. A postcode match is returned in either of these cases:

- You specified to match to postal code centroids. The resulting point is located at the postal code centroid with the following possible accuracy levels.
- There is no street level close match and you specified to fall back to postal code centroid.

Table 107: Postal (Z) Result Codes

Z Result Code	Description
Z1	Postal Code centroid match.
Z3	Full postal code centroid match. For Canada, this is an FSALDU centroid.

Postal level geocoded candidates return a result code beginning with the letter Z. Geocode Address World can generate a Z1 result code. Country-specific geocoders can often generate more accurate postcode results (with Z2 or Z3 result codes).

International Geographic Geocoding Result Codes (G Codes)

Geographic level geocoded candidates return a result code beginning with the letter G. The numbers following the G in the result code provides more detailed information on the accuracy of the candidate.

Table 108: Geographic (G) Result Codes

G Result Code	Description
G1	State or province centroid. match.
G2	County (district or region) centroid match.
G3	City or town (municipality) centroid match.
G4	Locality (village, suburb, or neighborhood) centroid match.

Reverse Geocoding Codes (R Codes)

Matches in the R category indicate that the record was matched by reverse geocoding. The second two characters of the R result code indicate the type of match found. R geocode results include an additional letter to indicate the dictionary from which the match was made.

Example reverse geocoding codes:

Table 109: Reverse Geocoding (R) Result Codes

Reverse Geocoding Code	Description
RS8A	Point/parcel level precision for reverse geocoding. Candidate returned from address dictionary.
RS5A	Interpolated street candidate for reverse geocoding. Candidate returned from address dictionary.
RS4A	Street centroid candidate for reverse geocoding. Candidate returned from address dictionary.

Non-match Codes

The following result codes indicate no match was made:

- **N**—No close match.
- **NX**—No close match for street intersections.
- **ND**—Spectrum™ Technology Platform could not find the geocoding database for the given postal code or municipality/state/province.

Encountering False Positives

What is a False-Positive?

To prevent the generation of address lists, the DPV and LACS^{Link} databases include false-positive records. False-positive records are artificially manufactured addresses that reside in a false-positive table. For each negative response that occurs in a DPV or LACS^{Link} query, a query is made to the false-positive table. A match to this table (called a false-positive match) disables your DPV or LACS^{Link} key. In batch processing the job that contains the violation will complete successfully but you will not be able to run any subsequent jobs that use DPV or LACS^{Link} until you report the violation and obtain a key to reactivate DPV or LACS^{Link}.

Note: The term "seed record violation" is also used to refer to encountering false positive records. The two terms mean the same thing.

Reporting DPV False-Positive Violations

Spectrum™ Technology Platform indicates a false-positive match via messages in the server log.

Client/server calls throw an exception if a false-positive match occurs. When a DPV fals positive record violation occurs, the server log will say:

```
WARN [Log] Seed record violation for S<ZIP, ZIP+4, Address, Unit> ERROR
[Log] Feature Disabled: DPU: DPV Seed Record Violation. Seed Code:
S<Address, ZIP, ZIP+4, Unit>
```

Note: If a DPV false positive record is found, the process() method (COM, C++, Java, and .NET) will throw an exception that the feature DPU has been disabled. In C, the processMessage() function will return a non-zero value.

You can report the violation and obtain a restart key by following these steps.

1. In your browser, go to http://<yourserver>:<port>/<product code>/dpv.jsp. For example, http://localhost:8080/unc/dpv.jsp for the Universal Addressing Module and http://localhost:8080/geostan/dpv.jsp for the Enterprise Geocoding Module.
2. Enter the mailer's information into each field. The number in parentheses after each field name indicates the maximum length of the field.
3. Click **Submit** when you're done. A **File Download** dialog will appear.
4. Click **Save** to save the file to your computer. A **Save As** dialog will appear.
5. Specify a file name and location on your local hard drive (for example `c:\DPVSeedFile.txt`) and click **Save**.
6. Go to **www.g1.com/support** and log in.
7. Click **DPV & LACS**^{Link} **False Positive**.
8. Follow the on-screen instructions to attach your seed file and obtain a restart key.

DPV False Positive Header File Layout

The USPS® has determined the required layout of the DPV false-positive header file, which is currently defined as a fixed-length file containing two or more 180-byte records. The first record must always be the header record, whose layout is shown below.

Table 110: DPV False-Positive Header Record Layout

Position	Length	Description	Format
1-40	40	Mailer's company name	Alphanumeric
41-98	58	Mailer's address line	Alphanumeric
99-126	28	Mailer's city name	Alphanumeric
127-128	2	Mailer's state abbreviation	Alphabetic
129-137	9	Mailer's 9-digit ZIP Code	Numeric
138-146	9	Total Records Processed	Numeric
147-155	9	Total Records DPV Matched	Numeric
156-164	9	Percent Match Rate to DSF	Numeric
165-173	9	Percent Match Rate to ZIP + 4®	Numeric
174-178	5	Number of ZIP Codes on file	Numeric
179-180	2	Number of False-Positives	Numeric

The trailer record contains information regarding the DPV false-positive match. There must be one trailer record added to the false-positive file for every DPV false-positive match. The layout is shown below.

Table 111: DPV False-Positive Trailer Record Layout

Position	Length	Description	Format
1-2	2	Street predirectional	Alphanumeric
3-30	28	Street name	Alphanumeric
31-34	4	Street suffix abbreviation	Alphanumeric
35-36	2	Street postdirectional	Alphanumeric
37-46	10	Address primary number	Alphanumeric
47-50	4	Address secondary abbreviation	Alphanumeric
51-58	8	Address secondary number	Numeric
59-63	5	Matched ZIP Code	Numeric
64-67	4	Matched ZIP + 4®	Numeric
68-180	113	Filler	Spaces

Reporting LACS/Link False-Positive Violations

Spectrum™ Technology Platform indicates a false-positive match via messages in the server log. Batch jobs will fail if a false-positive match occurs and client/server calls will throw an exception.

Note: The term "seed record violation" is also used to refer to encountering false positive records. The two terms mean the same thing.

When a false positive record is encountered, the server log will say:

```
2005-05-06 17:05:38,978 WARN [com.g1.component.ValidateAddress] Seed record
 violation for RR 2 28562  31373
2005-05-06 17:05:38,978 ERROR [com.g1.component.ValidateAddress] Feature
Disabled: LLU: LACS Seed Record Violation. Seed Code: 28562  31373
2005-05-06 17:05:38,978 ERROR [com.g1.dcg.gateway.Gateway] Gateway
exception: com.g1.dcg.stage.StageException:
com.g1.dcg.component.ComponentException: Feature Disabled: LLU
2005-05-06 17:06:30,291 ERROR
[com.pb.spectrum.platform.server.runtime.core.license.impl.policy.Policy]
 Feature LACSLink Real- time is disabled.
```

Note: If a LACS^{Link} false positive record is found, the process() method (COM, C++, Java, and .NET) will throw an exception that the feature LLU has been disabled. In C, the processMessage() function will return a non-zero value.

1. In your browser, go to http://<ServerName>:<port>/<product code>/lacslink.jsp. For example, http://localhost:8080/unc/lacslink.jsp for the Universal Addressing Module and http://localhost:8080/geostan/lacslink.jsp for the Enterprise Geocoding Module.
2. Enter the mailer's information into each field. The number in parentheses after the field name indicates the maximum length of the field. Click **Submit** when you're done. A **File Download** dialog will appear.
3. Click **Save** to save the file to your computer. A **Save As** dialog will appear.
4. Specify a file name and location on your local hard drive (for example `c:\lacslink.txt`) and click **Save**.
5. Go to **www.g1.com/support** and log in.
6. Click **DPV & LACS^{Link} False Positive**.
7. Follow the on-screen instructions to attach your seed file and obtain a restart key.

Enterprise Tax Module

Enterprise Tax Module

The Enterprise Tax Module (ETM) determines the tax jurisdiction for an address. The Enterprise Tax Module takes an address, standardizes it, and then matches the address to an exact physical location, returning latitude/longitude coordinates with the correct place code for the address. This solution greatly reduces the inaccuracies associated with 9-digit and 5-digit ZIP Code-based matching. The Enterprise Tax Module can also calculate latitude/longitude coordinates for individual address locations, including the use of interpolation and offset.

For companies that deliver goods or services to locations that don't have a recognizable address, the Enterprise Tax Module can accept latitude/longitude coordinates as input to return the tax jurisdictions and corresponding GeoTAX Key values. Examples of this type of location include bill boards, street lights, communications towers or new housing developments.

The Enterprise Tax Module uses a database of tax jurisdictions provided by TomTom. This data, which is collected through an ongoing research program and updated regularly, provides current jurisdictional boundary information down to the municipal and special tax district levels.

In addition to determining the tax jurisdiction for an address or location, with the optional Pitney Bowes Software Sales and Use Tax Rate file, the Enterprise Tax Module can return sales and use tax rates for each of the assigned tax jurisdictions as well as the total tax rate for the assigned locations. There is also the option to use other software, such as Vertex or Taxware, to cross-reference and supply tax rate data.

Note: The Enterprise Tax Module processes only U.S. addresses.

Enterprise Tax Components

The Enterprise Tax Module consists of the following components.

- **AssignGeoTAXInfo** - Takes an input address and returns census, latitude/longitude, and tax information about the address. AssignGeoTAXInfo utilizes Pitney Bowes Software's GeoTAX technology.
- **CalculateDistance** - Takes two latitude/longitude coordinates as input and computes and returns the distance between the coordinates.
- **ReverseGeoTAXInfo Lookup** - Takes an input latitude/longitude coordinate and returns census and tax information about the location. This service will not return an address for the location.

Enterprise Tax Databases

The Enterprise Tax Module provides you with several different databases, along with the ability to include additional databases to match against your input addresses or geographic coordinates.

Table 112: Enterprise Tax Module Databases

Database Name & Description	Required or Optional	Supplier
GeoTAX/Enterprise Tax Premium Master Files The master files are the main data files used by the Enterprise Tax module. They identify all geographic components associated with a street address, such as the tax jurisdictions, latitude/longitude, census tract, and block group. These files, at over two gigabytes of data, are significantly larger than the postal file, but provides the greatest coding accuracy.	Required	Pitney Bowes Software monthly/quarterly subscription

Database Name & Description	Required or Optional	Supplier
Note: The **GeoTAX/Enterprise Tax Premium Database** is a required data set and is the master file for the Enterprise Tax Premium Module software. This data is not compatible with any previously released version of the Enterprise Tax Module. The master files are available for download from the technical support web site, **www.g1.com/support**.		
State-Supplied Files State-supplied files are provided by individual state governments that the Enterprise Tax Module uses to override results from the master files. The Enterprise Tax Module provides you with the ability to override, at the state level, match results based upon information supplied by the states. By matching to the state-supplied files, you can remain compliant with tax jurisdiction assignment requirements mandated by new federal and state laws, such as the Mobile Telecommunications Sourcing Act and the Florida state Communications Services Tax Simplification Law. Currently, there are two file formats supported in the Enterprise Tax Module: the Florida-native format, and the national TS-158 format (ANSI Transaction Set No. 158). The state of Florida provides address files in both the TS-158 and its own native format. The state of Washington provides address data in the TS-158 format. **Note:** This database option may not be available to all Enterprise Tax users. Individual states may restrict the use of state-supplied address files to licensed communications carriers or other business entities registered with the individual state. The Enterprise Tax Module first attempts to match to the state database. If the Enterprise Tax Module cannot find a state match, it attempts a match to the GeoTAX Auxiliary file, if loaded, then to the master files.	Optional	State Governments
GeoTAX Auxiliary File The GeoTAX Auxiliary file contains new addresses that have not yet been added to the Master File. It provides the most up-to-date address data possible.	Optional	Pitney Bowes Software monthly subscription
Landmark Auxiliary File The Landmark Auxiliary File is a user-defined file that allows you to specify customized address information in your input records. The recommended primary use of this file to match to your company's non-address locations such as well heads, transmission towers or any other descriptive location. The latitude/longitude that is part of the input needed to build this file allows companies to automatically keep track of any jurisdictional changes that affect these unique locations. Matching to this file	Optional	User-defined

Database Name & Description	Required or Optional	Supplier
requires that the input record information match exactly to the Landmark file contents. This also applies to street records if you choose to enter them in the Landmark file. **Note:** For more information, see **Creating a Landmark Auxiliary File** on page 437 .		
User Auxiliary File User Auxiliary files are user-defined files that the Enterprise Tax Module uses to override results from the master files in street-level matching. If you have data that is more current than that in the master files, you can enter the new data into the auxiliary file and use it for your address matching. The Enterprise Tax Module returns matches made with a code that signifies the answer came from the auxiliary file. You can also return user-defined data from the auxiliary file with the match. **Note:** For more information, see **Creating a User-Defined Auxiliary File**.	Optional	User-defined
Boundary Files Boundary files provide additional data about locations of special tax districts: Special Purpose Tax Districts (SPD), Insurance Premium Tax Districts (IPDs), Payroll Tax Districts (PAYs), Personal Property Tax Districts (PTDs) and User-Defined Boundary (USR) . • The **Special Purpose District file** (SPD.txb) provides you with return data on special purpose tax districts. Special purpose tax districts include such districts as Regional Transit Areas and Metropolitan Football districts. • The **Insurance Premium District file** (IPD.txb) is used by the insurance industry to determine sales tax on insurance premiums written in some states. This file allows insurers to correctly determine the rate due on each insurance policy. Boundaries vary by state and are based on fire and police district and municipal boundaries. • The **Payroll Tax District (PAY) file** (PAY.txb) can help your company comply with state legislation that requires employers to deduct taxes from employee paychecks for special districts, such as taxes for emergency municipal services districts. • The **Personal Property Tax District** file (PTD.txb) provides your company with a complete solution for accurate, automated asset collection and jurisdiction assignment. Pitney Bowes Software provides you with the appropriate boundary file on separate media if you license any of the optional files. **Note:** The Enterprise Tax Module only uses one boundary file at a time. For more information, see the configuration options in **AssignGeoTAXInfo** on page 358.	Optional	Pitney Bowes Software
User-Defined Boundary File	Optional	User-created

Database Name & Description	Required or Optional	Supplier
A user-defined boundary file (usr.txb) is a file that you create to represent polygons that you want to match against, such as sales territories, insurance rating territories, or any geographic areas that are of interest to you. **Note:** For more information, see **Creating a User-Defined Boundary File** on page 443.		
Pitney Bowes Software Sales and Use Tax Rate File You can optionally license the Pitney Bowes Software Sales and Use Tax Rate file to supply sales and/or use tax rate data for the general, automotive, construction or medical industries. Sales and use tax rates can be returned for each of the assigned tax jurisdictions as well as the total tax rate for the assigned locations.	Optional	Pitney Bowes Software
Sales Tax Cross-Reference Files Sales tax cross-reference files allow you to use the Enterprise Tax Module to determine tax jurisdictions for a given address, then use third-party software to determine the sales tax rates for those jurisdictions. The cross-reference files combine the U.S. Government Federal Information Processing Standards (FIPS) codes with the proprietary geocodes used by tax software from third parties. **Note:** When using a cross-reference file, the SPD file must also be loaded. • **Vertex Files—**The Vertex files enable you to integrate the Enterprise Tax Module with tax compliance software from Vertex, Inc. With this file, the Enterprise Tax Module component AssignGeoTAXInfo can return the nine-digit Vertex jurisdiction code for an address. You can then match these codes to the Vertex tax tables, which tell you the tax rate for each jurisdiction. To use the Vertex file you must have either the MatchMaster file from Vertex or the PBS Vertex file from Pitney Bowes Software to build the cross reference. • **Taxware Files—**The Taxware files enable you to integrate the Enterprise Tax Module with tax compliance software from Taxware. With this file, the Enterprise Tax Module component AssignGeoTAXInfo can return the Taxware jurisdiction code for an address. You can then match these codes to the Taxware tax tables, which tell you the tax rate for each jurisdiction. The Taxware files are supplied by Pitney Bowes Software and support both the SUT and TWE Taxware versions.	Optional	Pitney Bowes Software
Payroll Tax Cross-Reference Files Payroll tax cross-reference files allow you to use third-party software to determine the payroll tax rates. The cross-reference files combine the U.S. Government Federal Information Processing Standards (FIPS) codes with the proprietary geocodes used by tax software from third parties. The Payroll	Optional	Pitney Bowes Software

Database Name & Description	Required or Optional	Supplier
System Tax Code file is a customized file that you build to return the payroll tax codes used by your payroll system. **Note:** For more information, see **Using a Payroll Tax Correspondence File** on page 434.		

Adding a Database Resource

Note: All ETM databases and cross-reference files must be downloaded to the same directory.

To add a database:

1. Under Tools, click the **Database Resources** icon.
2. Click the **Add...** button.
   The **Add Database Resource** window displays. Using this window, you have the option of adding one or more databases at a time.
3. Enter a name in the **Name** textfield to reference the database(s) selected.
4. If you are adding a State supplied file or a Sales tax cross-reference file, select a radio button option, then enter the path and filename of your database in the textfield or use the **...** button to navigate to your database.
5. For the required GeoTAX Master files and all other databases, enter the path and filename of your database in the textfield or use the **...** button to navigate to your database.
6. Click **OK** to complete the database load process.

Database Search Order

Enterprise Tax Module is able to process addresses using multiple databases at the same time. This allows you to find the best possible match from a variety of data sources and types of data (point as well as street segment data).

Enterprise Tax Module processes these multiple data sources using a default search order. When Enterprise Tax Module matches an address exactly, it stops searching rather than continuing the search in additional databases. This saves on processing time. When an exact match is not found, Enterprise Tax Module continues searching all of the available data sources for candidate address matches. The candidates are then scored and the highest scoring match from all of the data sources is returned as the match. If multiple candidates receive an identical score when matched to either the Points or Streets files, you have the option to set what results should be returned.

The default search order for Enterprise Tax Module is:

- Landmark Auxiliary file
- User Auxiliary file
- State-supplied file
- GeoTAX Auxiliary file
- Points file, if licensed
- Streets file

AssignGeoTAXInfo

AssignGeoTAXInfo identifies the tax districts that apply to a given address. Specifically, AssignGeoTAXInfo returns this information about an address:

- Latitude/longitude coordinates
- FIPS state codes and county codes
- County names
- MCD/CCD codes and names

- CBSA/CSA codes and names
- Place codes and names
- Incorporated or unincorporated status codes
- Cross-reference tax keys
- Result indicators
- Optionally, the relationship of an address to user-defined polygons

AssignGeoTAXInfo optionally includes enhanced tax jurisdiction information for an address, including:

- **Insurance premium districts**—Areas designated for the collection of taxes imposed on insurance policy premiums, based on the policy holder's address. Insurance premium districts are created by state governments.
- **Payroll tax districts**—Areas designated for the collection of taxes imposed on employers to support state or local government facilities and services, based on the employee's and/or employer's address. Examples include taxes collected for districts to pay for schools, police, or other services. Payroll tax districts are created by state or local governments.
- **Payroll system tax codes**—Codes that represent specific jurisdictions that collect payroll tax. Using payroll system tax codes has advantages over using the payroll tax district information returned by Assign GeoTAX Info:
  - AssignGeoTAXInfo uses an additional database to determine payroll tax codes, resulting in more accurate payroll tax determination.
  - Many payroll systems use specific codes to determine withholding amounts. Since you can customize the payroll tax codes returned by AssignGeoTAXInfo, you can set up a process where AssignGeoTAXInfo returns the exact payroll tax codes required by your payroll system, instead of returning jurisdictional IDs that must then be translated into the codes used by your system.
- **Property tax districts**—Areas designated for the collection of taxes imposed on property owners to support local government facilities and services, based on the property's location. Examples include taxes collected for districts to pay for schools, police, or other services. Property tax districts are created by local governments.
- **Special purpose tax districts**—Areas designated for the collection of taxes imposed on residents to support specialized services for residents of the district, based on the resident's address. Examples include services such as sewer service, transit service, or water resources. Special purpose tax districts are created by legislative action, court action, or public referendums. This optional information requires the use of boundary files which require an additional license. Contact your Pitney Bowes Software sales representative for more information.
- **Sales and Use Tax Rates**—Using the optional Pitney Bowes Software Sales and Use Tax Rate file, AssignGeoTAXInfo can return sales and use tax rates for each of the assigned tax jurisdictions as well as the total tax rate for the assigned locations.

AssignGeoTAXInfo is part of the Enterprise Tax Module.

Determining Tax Rates with AssignGeoTAXInfo

AssignGeoTAXInfo determines tax jurisdictions that apply to a given location. There are several options for determining tax rates for a location: use the optional Pitney Bowes Software Sales and Use Tax Rate file, use other software, such as Vertex or Taxware, or develop a custom process.

- Using Pitney Bowes Software Sales and Use Tax Rate file
  a) To include tax rate data in the output, set **GeoTAXOutputRecordType = B**.
  b) To select the tax rate type, set **TaxRate** to one of the following:

     G - Return the General sales and use tax rates.

     A - Return the Automotive sales and use tax rates.

     C - Return the Construction sales and use tax rates.

     M - Return the Medical sales and use tax rates.

- Using Vertex or Taxware Software
  a) To include Tax Jurisdiction fields in the output, which includes the GeoTAXKey field that contains the Vertex or Taxware code, set **GeoTAXOutputRecordType = T**.

b) To select the vendor information, set **TaxKey** to one of the following:

T - Return the Taxware jurisdiction code for the address.

Y - Return the Vertex jurisdiction code for the address. Select this option if you obtained a MatchMaster file from Vertex.

V - Return the Vertex jurisdiction code for the address. Select this option if you obtained a Vertex file from Pitney Bowes Software.

- Using a Custom Process

  If your organization has defined custom tax codes, you can use the jurisdiction information returned by AssignGeoTAXInfo to look up the tax jurisdictions in your custom tables. To do this, you need to configure AssignGeoTAXInfo to return the following fields:

  - StateCode
  - County.Code
  - GNISCode or Place.Code
  - SPDn.DistrictCode

  To include these fields in the output,

  a) To include Tax Jurisdiction fields in the output, set **GeoTAXOutputRecordType** = **T**.
  b) To include Special Purpose Districts fields in the output, set **GeoTAXOutputRecordType** = **S**.

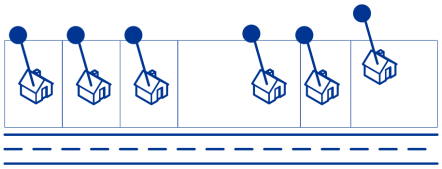
Address Matching

AssignGeoTAXInfo takes a house address and matches it to the correct street segment. Using the house number, it determines the side of the street on which the house resides (usually based on an odd-even division). Knowing the correct side of the street is important because streets are often the boundaries between municipalities or other adjoining jurisdictions that may have different tax rates. Thus, two addresses on the same street, in the same city, within the same ZIP Code could exist in different jurisdictions and have different tax rates.

After AssignGeoTAXInfo knows the correct street segment, it determines the actual physical location of the house based on known latitudes/longitudes and other geographic data in the street segment database. AssignGeoTAXInfo returns the coordinates of the house, along with other data.

Point-level matching

Point-level matching locates the center of the actual building footprint or parcel. This is the most accurate type of geocode.



If you are licensed for the point-level data option, you do not need to execute any additional initialization or setup for your system to begin using the point-level data. The Enterprise Tax Module automatically processes your address lists through the point-level data.

When processing address lists, the Enterprise Tax Module first searches for a match in the point-level data. If it cannot find an exact match in the point-level data, the Enterprise Tax Module continues searching for a better match in the street network data. The Enterprise Tax Module returns the best match found, with preference given to matches from the point-level dataset.

Note: Point data must be used in conjunction with a standard street data set.

Buffering

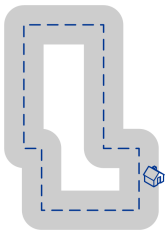
Use buffering to define areas that are close to the edges of a polygon, line, or point.

Buffered Polygon (zone)

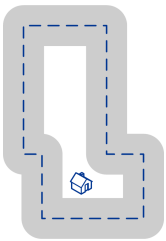
Buffered Line (corridor)

Buffered Point (circle)

For example, if you work for an insurance company you may want to know if a potential customer's house is within 500 feet of a flood plain so that you can suggest that they buy flood insurance even though they are not actually within the flood plain. The following illustration shows this scenario using a buffered polygon. The dotted line indicates the boundary of the flood plain and the shaded area shows a 500-foot buffer zone around the boundary.

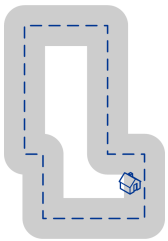


The buffer area extends on both sides of the boundary (inside and outside). When you use buffering, the output field BufferRelation indicates whether or not the point is in the buffered zone, and whether the point is inside or outside of the polygon, as shown in the following illustrations.



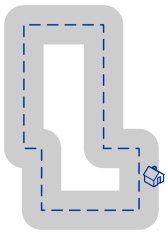
The point is inside the polygon and not in the buffer area.

The output field BufferRelation will contain "P".



The point is inside the polygon and in the buffer area.

The output field BufferRelation will contain "I".



The point is outside the polygon but in the buffer area.

The output field BufferRelation will contain "B".

Specify the size of polygon buffers using the BufferWidth input field to set it on a record-by-record basis and the DefaultBufferWidth option to set a default polygon buffer width for the job.

Input

The following table provides information on the format of AssignGeoTAXInfo input.

Note: Specify input using the DataTable class. For information on the DataTable class, see the "API Fundamentals" section.

Table 113: AssignGeoTAXInfo Input Data

columnName	Format	Description
AddressLine1	String [100]	First address line
AddressLine2	String [100]	Second address line

columnName	Format	Description
AddressLine3	String [100]	Third address line
AddressLine4	String [100]	Fourth address line
BufferWidth	String [10]	Specifies the width of the polygon buffers to use for Boundary File processing. The buffer width is used to determine if a point is close to the edge of a polygon. The output field BufferRelation indicates whether or not the point is within the polygon's buffer area. For more information, see **Buffering** on page 360. This field overrides the value specified in the DefaultBufferWidth option. Specify the border width in the units specified by the DistanceUnits option. If you do not specify a buffer width in this input field, the default is used.
CandidateID	String [1]	For a multi-match outcome, indicates which match result should be returned; valid values are 1-9.
City	String [50]	City name
Country	String [var]	The country where the address resides. The data you enter in this field has no impact on processing. It is simply passed through to output. **Note:** AssignGeoTAXInfo only supports US addresses.
FirmName	String [var]	Company or firm name
PostalCode	String [9]	Nine-digit ZIP Code
StateProvince	String [50]	The state where the address resides. The data you enter in this field has no impact on processing. It is simply passed through to output.
UserBufferWidth	Long [10]	Specifies the width of the polygon buffers to use for User-Defined Boundary File processing. The buffer width is used to determine if a point is close to the edge of a polygon. The output field BufferRelation indicates whether or not the point is within the polygon's buffer area. For more information, see **Buffering** on page 360. This field overrides the value specified in the DefaultBufferWidth option. Specify the border width in the units specified by the DistanceUnits option. If you do not specify a buffer width in this input field, the default is used.

Matching Options

Matching options control the address search methodology and match results handling returned by AssignGeoTAXInfo. **Table 114: AssignGeoTAXInfo Matching Options** on page 363 lists the address search and matching options.

Table 114: AssignGeoTAXInfo Matching Options

optionName	Description
Optional files: The following options enable the database resource(s) to use in the search process.	
UseGeoTaxAuxiliaryFile	Specifies whether or not AssignGeoTAXInfo should attempt a match to the GeoTAX Auxiliary file. The GeoTAX Auxiliary file contains new addresses that have not yet been added to the Master File. **Y** Use the GeoTAX Auxiliary file for matching. **N** Do not use the GeoTAX Auxiliary file for matching. (default)
UseAuxiliaryFile	Specifies whether or not AssignGeoTAXInfo should attempt a match to a User Auxiliary file. User Auxiliary files are user-defined files that the Enterprise Tax Module uses to override results from the master files in street-level matching. **Y** Use the User Auxiliary file for matching. **N** Do not use the User Auxiliary file for matching. (default)
UseStateProvidedFile	Specifies whether or not AssignGeoTAXInfo should attempt a match to the state-supplied file. Use this option in combination with FileSearchOrder to specify a state-supplied file to use. State-supplied files are provided by individual state governments. By matching to the state-supplied files, you can remain compliant with tax jurisdiction assignment requirements mandated by new federal and state laws, such as the Mobile Telecommunications Sourcing Act and the Florida state Communications Services Tax Simplification Law. There are two supported file formats: the Florida-native format and the national TS-158 format (ANSI Transaction Set No. 158). The state of Florida provides address files in both the TS-158 and its own native format. The state of Washington provides address data in the TS-158 format. Assign GeoTAX Info attempts to match to the state supplied file first. If a state match cannot be found, it attempts a match to the master files. **Y** Use the State-supplied file for matching. **N** Do not use the State-supplied file for matching. (default) **Note:** You must install the appropriate State-supplied file to use these options. For instructions, see the *Spectrum™ Technology Platform Installation Guide*.
FileSearchOrder	Specifies which state-supplied file to use. This option only takes effect if you specify UseStateProvidedFile=Y. One of the following: **FLOnly** Use only the Florida-native formatted file.

optionName	Description
	TSOnly Use only the TS-158 formatted file.
UseRelaxedSecondaryMatching	Specifies whether or not AssignGeoTAXInfo matches input addresses with secondary information to records without secondary information. This option applies only to Florida-native files. **Y** Use relaxed secondary matching. **N** Do not use relaxed secondary matching. (default)
Address Searching and Matching Options: The following options can be enabled for use in the address search and match processes.	
GsMatchMode	Match modes determine the leniency used to make a match between your input and the reference database. Select a match mode based on the quality of your input and your desired output. For example, if you have an input database that is prone to errors, you may want to select the relaxed match mode. **0 - Exact** Requires a very tight match. This restrictive mode generates the fewest match candidates, which decreases the processing time. When using this mode, ensure that your input is very clean; free of misspellings and incomplete addresses. **1 - Close** Requires a close match and generates a moderate number of match candidates. (default) **2 - Relaxed** Allows a loose match and generates the most match candidates, which increases the processing time and results in more multiple matches. Use this mode if you are not confident that your input is clean; free of misspellings and incomplete addresses. This is the only mode that does not respect the street parity when making an address match.
GsSearchArea	The search area options allow for searching the address' finance area or an expanded area specified by distance. **1** Searches the entire finance area for a match. A finance area is a region defined by the U.S. Postal Service and typically consists of a set of contiguous ZIP Codes.(default) **2** Searches the area specified by the radius in miles. The search area can be extended up to a 99-mile radius from the centroid of the input ZIP Code to assist in finding a match when the input address contains limited or inaccurate city or ZIP Code information. The expanded area is confined to within the state's borders.
GsSearchRadius	Radius for search area. **1-99 miles** Search radius. (default = 25 miles)
GsEnableFirstLetterExpanded	Looks for the correct first letter of a street address if the first letter is missing or incorrect. The Enterprise Tax Module searches through the alphabet looking for possible correct first letters to complete the street address.

optionName	Description
	Note: This feature is disabled by default and cannot be enabled in Exact mode. **Y** Enable first letter change matches. **N** Do not allow first letter change matches. (default)
GsEnableRangedAddress	Matches to a house range input. Some business locations are identified by address ranges. For example, a shopping plaza could be addressed as 10-12 Front St. - this is how business mail is typically addressed to such a business location. When this feature is enabled, the address range is geocoded to the interpolated mid-point of the range. **Y** Allow address range matches. **N** Do not allow address range matches. (default)
GsAlternateLookup	This option allows specifying the preferred way to match when both an address and firm name are provided. The matching method can be set to match to the address rather than the firm name or vice versa. If neither are specified, the default matching method is to match to the address line only. **1** Searches for street name, but if there isn't a match, will use the firm name. **2** Looks up the firm name, but if there isn't a match, will use the street name. **3** Searches only street records. (default)
GsMultiMatchResolution	A multi-match occurs when multiple equally-scored matches are found in either the Points or Streets files and cannot be resolved to a single best candidate. There are several choices for handling a multi-match outcome: **N** No matches are returned. (default) **R** Return the first match candidate in the list. **A** The information for all the match candidates is returned.

Specifying Default State-Supplied File Options

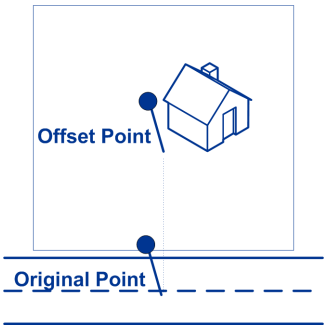
If you use the Spectrum™ Technology Platform API, note that the value you specify in the Management Console for the **State supplied file** field controls the default settings for three AssignGeoTAXInfo API options: UseStateProvidedFile, UseRelaxedSecondary, and FileSearchOrder. The following table show how each value in the **State supplied file** field affects these three options.

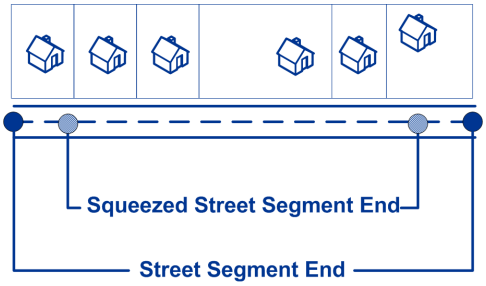
"State supplied file" value	UseStateProvidedFile	FileSearchOrder	UseRelaxedSecondary
None	N	N/A	N
Florida-native	Y	FSOnly	N
Florida-native with relaxed secondary matching	Y	FSOnly	Y
TS-158	Y	TSOnly	N

Geocoding Options

Geocoding is the process of determining the latitude/longitude coordinates of a given address. Address coordinates are used as the basis for determining the tax jurisdictions for an address. Geocoding options control how AssignGeoTAXInfo determines address latitude/longitude coordinates.

Table 115: AssignGeoTAXInfo Geocoding Options

optionName	Description
Database.GTX	Select the database resource to use in the geocoding process.
Latitude/Longitude placement: The following options can be set for the geocode result.	
LatLongOffset	Indicates the offset distance in feet from the street center line. The offset distance is used in street-level geocoding to prevent the geocode from being in the middle of a street. It compensates for the fact that street-level geocoding returns a latitude and longitude point in the center of the street where the address is located. Since the building represented by an address is not on the street itself, you do not want the geocode for an address to be a point on the street. Instead, you want the geocode to represent the location of the building which sits next to the street. For example, an offset of 50 feet means that the geocode will represent a point 50 feet back from the center of the street. The distance is calculated perpendicular to the portion of the street segment for the address. Offset is also used to prevent addresses across the street from each other from being given the same point. The following diagram shows an offset point in relation to the original point.  **0** No offset. (default) **20** Twenty feet offset from street center line. **40** Forty feet offset from street center line. **60** Sixty feet offset from street center line.
Squeeze	Specifies if AssignGeoTAXInfo should squeeze the street end points when determining the geocode of an address in street-level matching. The squeeze value is 5% (2.5% from each endpoint). The following diagram compares the end points of a street segment to the squeezed end points of a street segment.

optionName	Description
	 **Y** Apply squeeze. **N** Do not apply squeeze. (default)
LatLongFormat	Indicates the desired format for returned latitude/longitude. Options include: **PreZero** Latitude/longitude in the following format: 090000000N180000000W. (default) **PreZeroDecimal** Latitude/longitude in the following format: 090.000000N180.000000W. **Decimal** Latitude/longitude in the following format: 90.000000-180.000000. **DecimalAssumed** Latitude/longitude in the following format: 90000000-180000000. **DegMinSec** Latitude/longitude in the following format: 90 00 00N180 00 00W.
Expanded Geocoding options: The following options enable additional geocoding functionality.	
GsEnableAddressPointInterpolation	Address point interpolation uses a patented process that improves upon regular street segment interpolation by inserting point data into the interpolation process. **Note:** This feature is only for use with point-level geocoding. A match is first attempted using the loaded points data. If an exact point match is found in the points data, then searching ceases and the point match is returned. If an exact point match was not found, the Enterprise Tax Module attempts to find high and low boundary address points to use for address point interpolation. **Y** Enable address point interpolation. **N** Disable address point interpolation. (default)
Minimum geocode quality:	
GsEnableGeographicFallback	The default search does not perform a search of geographic centroids. When enabled, the Geographic Fallback feature locates the first city, county and/or state centroid, and then matches from the set of possible matches found. **Y** If a definitive match cannot be made, then return the next higher level geographic centroid. **N** Disable geographic fallback feature. (default)

optionName	Description
GsEnableStreetCentroid	If an input street address cannot be found using the street number and name, the Enterprise Tax Module then searches the input ZIP Code or city/state for the closest match. If the Enterprise Tax Module is able to locate the street, it returns a geocode along the matched street segment rather than the geocode for the entered ZIP Code or ZIP + 4. When using street locator geocoding, if no exact matching house number is found, a match code of either E029 (no matching range, single street segment found), or E030 (no matching range, multiple street segment) returns. For example, if you enter Main St and there are both an E Main St and a W Main St within the input ZIP Code, then an E030 returns and the location code returned is reflective of the input ZIP Code. The location code returned begins with a 'C' when matched to a single street segment, indicated by E029. The Enterprise Tax Module does not change the street name on the output address. **Y** If a street or point match cannot be made, then return a street level centroid. **N** Do not return a street level centroid if a match cannot be made. (default)
Boundary matching: The following options can be set when matching to a boundary file such as SPD, IPD, PAY, Place and MCD or user-defined.	
DistanceUnits	Specifies the units in which to measure distance. One of the following: **Miles** Distances are measured in miles. **Km** Distances are measured in kilometers. **Feet** Distances are measured in feet. (default) **Meters** Distances are measured in meters.
DefaultBufferWidth	Specifies the buffer width to use for tax district boundary files. The tax district boundary files are the Special Purpose District (SPD) file, the Insurance Premium District (IPD) file, the Payroll Tax District (PAY) file, the Personal Property Tax District (PTD) file and Place and MCD files. The default buffer width that you specify here can be overridden on a record-by-record basis using the BufferWidth input field. For more information on buffers, see **Buffering** on page 360.
DefaultUserBufferWidth	Specifies the buffer width to use for user-defined boundary files. Specify the distance in the units of measurement specified in the **Distance units** option. For information on buffers, see **Buffering** on page 360. The default buffer width that you specify here can be overridden on a record-by-record basis using the BufferWidth input field. **Note:** To use buffers, the user-defined boundary file must support buffers.

Output Data Options

Data options control the data returned by AssignGeoTAXInfo. **Table 116: AssignGeoTAXInfo Data Options** on page 369 lists the output data options.

Table 116: AssignGeoTAXInfo Data Options

optionName	Description
GeoTAXOutputRecordType	Select one or more of the following to obtain the type of data you want returned. AssignGeoTAXInfo groups the output fields into record types. If you do not want all of the fields in a record type returned, do not select the check box, and list only those fields you want returned in Extra Output Fields. • **C**—Census • **L**—Latitude/Longitude • **T**—Tax Jurisdiction • **U**—User-defined boundary file • **W**—Payroll System Tax Codes • **X**—Auxiliary File • **B**—PB Software Sales and Use Tax Rate file You can also specify one, and only one, of the following: **I** Insurance Premium Tax District (IPD) **P** Property Tax District (PTD) **R** Payroll Tax District (PAY) **S** Special Purpose Tax District (SPD) For a description of the fields in each output group, see **Output** on page 370. **Note:** If you specify `W`, also specify `R` to obtain the best payroll system tax code match possible.
TaxKey	If you integrate AssignGeoTAXInfo with third-party tax compliance software from Vertex or Taxware, select which vendor you use. This controls the value returned in the GeoTAXKey output field. One of the following: **T** Return the Taxware jurisdiction code for the address. **Y** Return the Vertex jurisdiction code for the address. Select this option if you obtained a MatchMaster file from Vertex. **V** Return the Vertex jurisdiction code for the address. Select this option if you obtained a Vertex file from Pitney Bowes Software. **N** Do not return either the Taxware or Vertex jurisdiction codes (default).
TaxRate	Indicates the sales and use tax rate type to return or None: **N** Do not return sales and use tax rates. (default) **G** Return the General sales and use tax rates.

optionName	Description
	A Return the Automotive sales and use tax rates. **C** Return the Construction sales and use tax rates. Return the Medical sales and use tax rates.
OutputFields	Indicates the individual output fields you want returned. You can use this field instead of the Output Record Type to limit the output to those fields that are important to your current data needs. For a list of the fields included in each data type, see **Output** on page 370.

Output Format

Output format options control how AssignGeoTAXInfo formats output data. The following table lists the output format options.

Table 117: AssignGeoTAXInfo Output Format Options

optionName	Description
OutputCasing	Specifies the casing of these output fields: County.Name, CBSA.Name, MCD.Name, Place.Name, IPDn.DistrictName, PAYn.DistrictName, SPDn.DistrictName, and PTCn.PayrollDescription. One of the following: **M** The output in mixed case (default). For example: Rensselaer. **U** The output in upper case. For example: RENSSELAER.

Output

Result Indicators

The following table lists the results returned from the address matching and geocoding process.

Table 118: Address Matching and Geocoding Status Output Fields

columnName	Max. Field Length (bytes)	Description
Confidence	4	Indicates the confidence in the output provided; from 0 to 100. The higher the score, the higher the confidence in the match. Calculated based on the match results for individual output fields, using the following algorithm: Census.MatchCode + LatLong.StreetMatchCode + LatLong.MatchCode

columnName	Max. Field Length (bytes)	Description

		The maximum confidence score is 100, so if this calculation results in a value greater than 100, the Confidence score is returned as 100.
		AssignGeoTAXInfo uses the following values:
		• Census.MatchCode
		  • A = 85
		  • G = 85
		  • S = 85
		  • U = 85
		  • 9 = 65
		  • 5 = 45
		  • null = 0

Confidence.SurfaceType	3	Indicates the confidence surface type. Setting a non-zero buffer width enables confidence generation. To determine a confidence level, a confidence surface is first generated. The confidence surface provides the smallest possible area wherein an address is likely to be located. **0** Undefined **1** The search failed - the address was not found. **2** Intersection confidence surface generated. **3** Interpolated street segment. **4** Point-level match. **5** State confidence surface generated. **6** County confidence surface generated. **7** City confidence surface generated. **8** Reserved **9** A ZIP Code confidence surface generated. **10** A ZIP+2 confidence surface generated. **11** A ZIP+4 confidence surface generated. **12** Reserved **13** A street centroid confidence surface generated.
AddressMatch.MatchCode	5	The Match Code indicates the portions of the address that matched or did not match to the reference file. For the list of match codes, see **Match Codes.**
AddressMatch.GenRC	2	General Return Code indicating the type of match. **5** ZIP Code match

columnName	Max. Field Length (bytes)	Description
		9 ZIP+4 Code match **A** User Auxiliary file match **C** Street Centroid match **F** Geographic Fallback match **G** State-supplied file match **I** Intersection match **L** Landmark Auxiliary file match **M** Multiple match (multi-match) **O** Input Latitude/Longitude coordinates match **P** Address point match **S** Street address match **X** Aborted processing or expired database **Blank** Did not match
AddressMatch.DataTypeName	20	Indicates the file from which the match was obtained. One of the following: • USPS • TIGER • TOMTOM - Streets • NAVTEQ - Streets • TOMTOM_POINT • CENTRUS_POINT • NAVTEQ_POINT • STATE_FILE • USER_AUXILIARY • LANDMARK_AUXILIARY
AddressMatch.NumCandidates	2	When there are multiple equally-scored matches, returns the number of multiple match candidates found.
AddressMatch.LocationCode	5	The Location Code indicates the methodology used to complete the geocode and may also provide some information about the quality of the geocode. For the list of location codes, see **Location Codes.**
LatLong.MatchCode	2	Latitude/Longitude General Return Code. Denotes the level for which the geocode was determined. **2** ZIP + 2 centroid **4** ZIP + 4 Code centroid **B** Block group centroid **C** City centroid **I** Intersection

columnName	Max. Field Length (bytes)	Description
		L Match using the Landmark Auxiliary file **O** Latitude/longitude was input **R** Address-level based on street address **S** State centroid **T** Census tract centroid **U** Address-level match using the GeoTAX Auxiliary Database **Z** ZIP Code centroid based on a five-digit ZIP code **null** No latitude/longitude determined If the **General Return Code** (see **Result Indicators**) is "**P**" (point match), then the following are possible values and have the following meaning: **0** Latitude/Longitude coordinates from User Dictionary. **2** Latitude/Longitude coordinates from Parcel Centroid. **4** Latitude/Longitude coordinates from Address Point. **5** Latitude/Longitude coordinates from Structure Centroid. **7** Latitude/Longitude coordinates from manually-placed Point. **8** Latitude/Longitude coordinates from Front Door Point. **9** Latitude/Longitude coordinates from Driveway Offset Point. **A** Latitude/Longitude coordinates from Street Access Point. **B** Latitude/Longitude coordinates from Base Parcel Point.
LatLong.StreetMatchCode	2	Output street address return code. **H** House number not found on street **L** Latitude/longitude not determined on auxiliary match **S** Street not found in ZIP Code **Z** ZIP Code not found in street address database **N** Street-level matching option not selected

columnName	Max. Field Length (bytes)	Description
		null The street was successfully matched

Auxiliary File

The following table lists the output fields that contain auxiliary file data. To include auxiliary file data in the output, set **GeoTAXOutputRecordType** = X. The following table lists the output fields that contain tax jurisdiction data.

Table 119: Auxiliary File Output Fields

columnName	Max. Field Length (bytes)	Description
AuxiliaryData.AuxiliaryFile	301	Data retrieved as a result of an auxiliary match from the user-defined area of the auxiliary file.
AuxiliaryData.StateFile	201	Data retrieved as a result of a state match. Data content and format vary depending on the state file used.

Census

The census output fields contains census information from the U.S. Census, including Minor Civil Divisions (MCDs) and Census County Division (CCD) names and codes. MCDs are the primary political or administrative divisions of a county, representing many kinds of legal entities with a variety of governmental and administrative functions. CCDs are established in states where there are no legally established MCDs. The Census Bureau recognizes MCDs in 28 states and has established CCDs in 21 states. The District of Columbia has no primary divisions, and the city of Washington, DC is considered equivalent to an MCD for data presentation purposes.

Census data also contains the Federal Information Processing Standards (FIPS) codes for each state and county. The FIPS State Code and the FIPS County Code are both used by the Census Bureau to identify these geographic units.

The following table lists the output fields that contain census data. To include census data in the output, set **GeoTAXOutputRecordType** = C.

Table 120: Census Data Output Fields

columnName	Max. Field Length (bytes)	Description
Census.Block	4	Census Block ID.
Census.BlockGroup	2	Census Block Group code.
Census.MatchCode	2	The level of match obtained against the databases.

columnName	Max. Field Length (bytes)	Description
		Note: This field is always included in the output regardless of whether or not you choose to include census data in the output. **A** Auxiliary street match **G** State file address match **S** Street address match **U** GeoTAX Auxiliary file match **9** ZIP + 4 Code level match **5** ZIP Code level match **null** Unsuccessful match
Census.MatchLevel	19	The level of match obtained against the databases. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output. **Auxiliary** Auxiliary street match **Aux2** GeoTAX Auxiliary file match **Street** Street address match **Gov** State file address match **ZIP+4** ZIP + 4 Code level match **ZIP** ZIP Code level match **null** No match
Census.Tract	7	Six-digit tract number extracted from the Census.BlockCode.
County.Code	4	Three-digit Federal Information Processing Standards (FIPS) county code extracted from the Census.BlockCode. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.
County.Name	26	Name of the county. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.
MCD.Code	6	Minor Civil Division/Census County Division (MCD/CCD) Code.
MCD.Name	41	Minor Civil Division/Census County Division (MCD/CCD) name.

columnName	Max. Field Length (bytes)	Description
MCD.PointStatus	2	An address can be compared to a Minor Civil Division (MCD)/county subdivision file (Cousub.txt). This output field returns the result for a comparison between the matched address point to the polygon defined by the Cousub.txb file. For more information on buffers, see **Buffering** on page 360. **Note:** The buffer distance for Cousub.txb is internally set to zero and cannot be modified. **P** The point is in the polygon. **I** The point is in the buffer area inside the polygon. **B** The point is in the buffer area and outside of the polygon. **blank** Polygon not found.
MCD.DistanceToBorder	10	Returns the distance in feet between the matched address point to the polygon defined by the Cousub.txb file.
MCD.Confidence	4	Provides the percentage overlap of the geocode to the MCD polygon layer. The returned percentage value describes the probability that the point falls in the MCD.
CBSA.Code	6	Core Based Statistical Area (CBSA) code.
CBSA.Name	76	Core Based Statistical Area (CBSA) name.
CBSA.MetroFlag	2	Indicates if the CBSA is a "Metropolitan Statistical Area" or a "Micropolitan Statistical Area". **Y** Metropolitan Statistical Area - A Core Based Statistical Area associated with at least one urbanized area that has a population of at least 50,000. The Metropolitan Statistical Area comprises the central county or counties containing the core, plus adjacent outlying counties having a high degree of social and economic integration with the central county as measured through commuting. **N** Micropolitan Statistical Area - A Core Based Statistical Area associated with at least one urban cluster that has a population of at least 10,000, but less than 50,000. The Micropolitan Statistical Area comprises the central county or counties containing the core, plus adjacent outlying counties having a high degree of social and economic integration with the central county as measured through commuting.
CBSAD.Code	6	Core Based Statistical Area Division (CBSAD) code.

columnName	Max. Field Length (bytes)	Description
CBSAD.Name	73	Core Based Statistical Area Division (CBSAD) name.
CSA.Code	4	Combined Statistical Area (CSA) code.
CSA.Name	78	Combined Statistical Area (CSA) name.
State.Abbreviation	3	Two-character state abbreviation. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output
StateCode	3	Two-digit Federal Information Processing Standards (FIPS) state code extracted from the Census.BlockCode. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.

Latitude/Longitude

The following table lists the output fields that contain latitude and longitude data. Latitude/Longitude data contains the coordinates for the address and additional information about how AssignGeoTAXInfo determined the latitude and longitude. To include latitude/longitude data in the output, set **GeoTAXOutputRecordType** = L.

Table 121: Latitude/Longitude Output Fields

columnName	Max. Field Length (bytes)	Description
Latitude	8	Seven-digit number in degrees and calculated to four decimal places (in the format you specified).
Latitude.Directional	2	Latitude directional. **N** North **S** South
LatLong	23	Returned latitude/longitude, in the format you specified (up to 22 alphanumeric characters).
LatLong.MatchCode	2	Latitude/Longitude General Return Code. Denotes the level for which the geocode was determined. **2** ZIP + 2 centroid **4** ZIP + 4 Code centroid **B** Block group centroid **C** City centroid **I** Intersection

columnName	Max. Field Length (bytes)	Description
		L Match using the Landmark Auxiliary file **O** Latitude/longitude was input **R** Address-level based on street address **S** State centroid **T** Census tract centroid **U** Address-level match using the GeoTAX Auxiliary Database **Z** ZIP Code centroid based on a five-digit ZIP code **null** No latitude/longitude determined If the **General Return Code** (see **Result Indicators**) is "**P**" (point match), then the following are possible values and have the following meaning: **0** Latitude/Longitude coordinates from User Dictionary. **2** Latitude/Longitude coordinates from Parcel Centroid. **4** Latitude/Longitude coordinates from Address Point. **5** Latitude/Longitude coordinates from Structure Centroid. **7** Latitude/Longitude coordinates from manually-placed Point. **8** Latitude/Longitude coordinates from Front Door Point. **9** Latitude/Longitude coordinates from Driveway Offset Point. **A** Latitude/Longitude coordinates from Street Access Point. **B** Latitude/Longitude coordinates from Base Parcel Point.
LatLong.MatchLevel	14	A description of the value returned in the LatLong.MatchCode field. **ZIP+2** ZIP + 2 centroid **ZIP+4** ZIP + 4 centroid **Block** Block group centroid **CityCentroid** City centroid **Intersection** Intersection match

columnName	Max. Field Length (bytes)	Description	
		LandmarkAux	Match using the Landmark Auxiliary file
		LatLonInput	Input Latitude/Longitude coordinates was used
		Rooftop	Exact address match
		StateCentroid	State centroid
		Tract	Census tract centroid
		Auxiliary	Address-level match using the GeoTAX Auxiliary Database
		ZIP	ZIP Code centroid
LatLong.StreetMatchCode	2	Output street address return code.	
		H	House number not found on street
		L	Latitude/longitude not determined on auxiliary match
		S	Street not found in ZIP Code
		Z	ZIP Code not found in street address database
		N	Street-level matching option not selected
		null	The street was successfully matched
LatLong.StreetMatchLevel		Street level match used to determine the latitude/longitude	
		FullMatch	Successful match
		HouseNotFound	House number not found on street
		LatLongNotFound	Latitude/longitude not determined on auxiliary match
		StreetNotFound	Street not found in ZIP Code
		ZipNotFound	ZIP Code not found in street address database
		NotUsed	Street-level matching option not selected
Longitude	8	Seven-digit number in degrees and calculated to four decimal places (in the format specified).	
Longitude.Directional	2	Longitude directional.	
		E	East
		W	West

Input Address

AssignGeoTAXInfo always returns the input address as part of the output. The input address fields are returned as input from the data. AssignGeoTAXInfo does not change these input values.

Table 122: Input Address Output Fields

columnName	Max. Field Length (bytes)	Description
AddressLine1	101	Input address line 1.
AddressLine2	101	Input address line 2.
AddressLine3	101	Input address line 3.
AddressLine4	101	Input address line 4.
City	51	Input address city.
Country	25	Input address country.
FirmName	101	Input address firm name.
PostalCode	10	Input address postal code
StateProvince	51	Input address state.

Payroll System Tax Code

The following table lists the output fields that contain Payroll System Tax Code (PTC) data. For more information on payroll tax districts, see **AssignGeoTAXInfo** on page 358. To include this data in the output, set **GeoTAXOutputRecordType** = W.

Note: AssignGeoTAXInfo returns up to six payroll tax codes per address.

Table 123: Payroll System Tax Code Output Fields

columnName	Max. Field Length (bytes)	Description
NumberPTCsFound	2	The number of payroll tax codes found for this address.
PTC*n*.MatchCode	2 per PTC	Indicates the level of match obtained for the address. In order from most specific match to least, the possible match codes are: **P** The address was matched to a specific Payroll District ID. This is the most specific match. **G** The address was matched to a GNIS Code. **F** The address was matched to a county's FIPS code. **S** The address was matched to a state's FIPS code. This is the least specific match.

columnName	Max. Field Length (bytes)	Description
PTC*n*.PayrollCode	16 per PTC	A code that represents a taxing authority in a payroll application. This is a user-defined code. The specific codes are determined by the payroll application that utilizes the data returned by AssignGeoTAXInfo.
PTC*n*.PayrollDescription	41 per PTC	A description of the purpose of this payroll code.
PTC*n*.PayrollFlag	7 per PTC	A user-defined flag from the PTC database.
StateCounty	33	The state abbreviation and county name.

Tax Jurisdiction

Tax jurisdiction data contains information about the "place" for the location. A "place" is a geographic area defined on the basis of population criteria that vary by state; or, an area recognized as significant because it is located in an incorporated municipality. Places are used to determine tax jurisdiction.

The following table lists the output fields that contain tax jurisdiction data. To include tax jurisdiction data in the output, set **GeoTAXOutputRecordType** = T.

Table 124: Tax Jurisdiction Output Fields

columnName	Max. Field Length (bytes)	Description
GeoTAXKey	10	The value in this field varies depending on the option you specified in the TaxKey option: If you specified `T`, GeoTAXKey contains the proprietary codes used in Taxware tax compliance software. You can use this code in your Taxware application to find out the tax rate for the jurisdiction. If you specified `Y` or `V`, GeoTAXKey contains the proprietary Vertex® jurisdiction code (comprised of a two-digit Vertex® state code, three-digit FIPS county code, and four-digit Vertex® city code). You can use this code in your Vertex® application to find out the tax rate for the jurisdiction.
GeoTAXKey.MatchCode	2	Return code denoting the level of match obtained against the Vertex or Taxware cross reference files. **E** Exact match using five fields: FIPS state code, FIPS county code, FIPS or GNIS place code, ZIP Code, and FIPS place name. **P** Partial match using four fields: FIPS state code, FIPS county code, FIPS or GNIS place code, and ZIP Code.

columnName	Max. Field Length (bytes)	Description
		A Alternate match using two fields: ZIP Code, FIPS place name. **N** Record is default coded based on valid state code. **null** No matching record found.
GeoTAXKey.MatchLevel	12	A description of the value returned in the GeoTAXKey.MatchCode field. **Exact** Exact match. See description in GeoTAXKey.MatchCode. **Partial** Partial match. See description in GeoTAXKey.MatchCode. **Alternate** Alternate match. See description in GeoTAXKey.MatchCode. **DefaultCode** Record is default coded. See description in GeoTAXKey.MatchCode. **NoMatch** No matching record found.
GNISCode	10	Unique nine-digit Geographic Names Information System (GNIS) code. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.
Place.ClassCode	3	Place class code. Place class codes are used to determine the proper taxing jurisdictions **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.Code	6	An identifier for a specific place. A "place" is a geographic area defined on the basis of population criteria that vary by state. Or, an area recognized as significant because it is located in an incorporated municipality. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.IncorporatedFlag	8	Indicates whether the address is located in an incorporated or unincorporated place. A "place" is a geographic area defined on the basis of population criteria that vary by state. Or, an area recognized as significant because it is located in an incorporated municipality.

columnName	Max. Field Length (bytes)	Description
		Note: This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output. **Inc** Incorporated place code. **Uninc** Unincorporated place code. **Unknown** Incorporation status unknown.
Place.LastAnnexedDate	8	Last annexed date, in the format MM/YYYY, representing the month and year of the most recent boundary change or the most recent available boundary information. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.LastUpdatedDate	8	Last updated date, in the format MM/YYYY, reflecting the month and year when TomTom updated the database to reflect attribute (name change, FIPS change, etc.) or boundary edits to the Place. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.LastVerifiedDate	8	Last verified date, in the format MM/YYYY, representing the month and year that TomTom verified municipality change information. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.Name	41	The name of the "place" where the address is located. A "place" is a geographic area defined on the basis of population criteria that vary by state. Or, an area recognized as significant because it is located in an incorporated municipality. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.PointStatus	2	Returns the result for a comparison between the matched address point to the polygon defined by the Place.txb file. For more information on buffers, see **Buffering** on page 360. **Note:** The buffer distance for Place.txb is internally set to zero and cannot be modified. **P** The point is in the polygon.

columnName	Max. Field Length (bytes)	Description
		I The point is in the buffer area inside the polygon. **B** The point is in the buffer area and outside of the polygon. **blank** Polygon not found. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.DistanceToBorder	10	Returns the distance between the matched address point to the polygon defined by the Place.txb file. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.Confidence	4	Provides the percentage overlap of the geocode to the Place polygon layer. The returned percentage value describes the probability that the point falls in the specified Place. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.

User-Defined Boundary File

The following table lists the output fields that contain data returned from user-defined boundary files. To include this data in the output, set **GeoTAXOutputRecordType** = U.

Note: AssignGeoTAXInfo can return up to 10 user-defined areas for each input address.

Table 125: Output Fields for User-Defined Boundary Files

columnName	Max. Field Length (bytes)	Description
NumberUserBoundariesFound	3	The number of user-defined polygons found for the address.
UserBoundary*n*.BoundaryDescription	51 per User Boundary	A description of the polygon.
UserBoundary*n*.BoundaryID	11 per User Boundary	The ID of the polygon as specified in the user-defined boundary file.

columnName	Max. Field Length (bytes)	Description
UserBoundary*n*.BufferRelation	2 per User Boundary	Indicates where in the polygon the address resides in relation to the edge of the area. One of the following: **P** The address is inside the polygon at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The address is inside the polygon but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The address is outside the polygon but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
UserBoundary*n*.DistanceToBorder	10 per User Boundary	Indicates the distance from the address to the border of the polygon. The distance is in the units specified by the option DistanceUnits.
UserBoundary*n*.SupplementalBoundaryID	11 per User Boundary	A supplemental ID as specified in the user-defined boundary file.
UserBoundary*n*.BoundaryConfidence	4 per User Boundary	Provides the percentage overlap of the geocode to the User-defined boundary polygon layer. The returned percentage value describes the probability that the point falls in the User-defined boundary area.

Insurance Premium Tax District

The following table lists the output fields that contain Insurance Premium Tax District (IPD) data. For more information on insurance premium tax districts, see **AssignGeoTAXInfo** on page 358. To include IPD data in the output, set **GeoTAXOutputRecordType** = I.

Note: AssignGeoTAXInfo returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 126: Insurance Premium Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberIPDsFound	3	The number of Insurance Premium Tax Districts found for the address

columnName	Max. Field Length (bytes)	Description
IPD*n*.BoundaryBuffer.BufferRelation	2 per IPD	Indicates where in the district the address resides in relation to the edge of the district. One of the following: **P** The address is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The address is inside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The address is outside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
IPD*n*.BoundaryBuffer.DistanceToBorder	10 per IPD	Indicates the distance from the address to the border of the district.
IPD*n*.BoundaryConfidence	4 per IPD	Provides the percentage overlap of the geocode to the IPD boundary polygon layer. The returned percentage value describes the probability that the point falls in the IPD boundary area.
IPD*n*.DistrictID	11 per IPD	IPD ID.
IPD*n*.DistrictName	61 per IPD	IPD name.
IPD*n*.DistrictType	7 per IPD	IPD district type.
IPD*n*.UpdateDate	7 per IPD	IPD update date (MMYYYY).
IPD*n*.VersionDate	7 per IPD	IPD compiled date (MMYYYY).
IPD*n*.Notes	21 per IPD	Tax code descriptions. For example: 01, 33, A, B
IPD*n*.ChangeDate	7 per IPD	IPD change date.
IPD*n*.EffectiveDate	7 per IPD	MMDDYY - Identifies when district becomes active - State supplied For example: 010108

columnName	Max. Field Length (bytes)	Description
IPD*n*.ExpirationDate	7 per IPD	MMDDYY - Identifies when district becomes inactive - State supplied For example: 063009
IPD*n*.FireRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.FireFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semi colon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.CasualtyRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.CasualtyFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.VehicleRate	21 per IPD	Format is dependent on associated flag Possible Values: .13, 15.00 or 3;7
IPD*n*.VehicleFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.MarineRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.MarineFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.HealthRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.HealthFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.LifeRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7

columnName	Max. Field Length (bytes)	Description
IPD*n*.LifeFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.OtherRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.OtherFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.MinimumRate	21 per IPD	Format is dependent on associated flag Possible Values: .13, 15.00 or 3;7
IPD*n*.MinimumFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"

Payroll Tax District

The following table lists the output fields that contain Payroll Tax District (PAY) data. For more information on payroll tax districts, see **AssignGeoTAXInfo** on page 358. To include this data in the output, set **GeoTAXOutputRecordType** = R.

Note: AssignGeoTAXInfo returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 127: Payroll Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberPAYsFound	3	Number of PAYs returned.
PAY*n*.BoundaryBuffer.BufferRelation	2 per PAY	Indicates where in the district the address resides in relation to the edge of the district. One of the following: **P** The address is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The address is inside the district but is close to the edge. This indicates that the address is in the

columnName	Max. Field Length (bytes)	Description
		buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The address is outside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
PAY*n*.BoundaryBuffer.DistanceToBorder	10 per PAY	Indicates the distance from the address to the border of the district. The distance is in the units specified by the option DistanceUnits.
PAY*n*.BoundaryConfidence	4 per PAY	Provides the percentage overlap of the geocode to the PAY boundary polygon layer. The returned percentage value describes the probability that the point falls in the PAY boundary area.
PAY*n*.DistrictID	11 per PAY	PAY district ID.
PAY*n*.DistrictName	61 per PAY	PAY district name.
PAY*n*.DistrictType	7 per PAY	PAY district type.
PAY*n*.ID	11 per PAY	PAY ID.
PAY*n*.MunicipalEMSTax	2 per PAY	PAY municipality emergency municipal services tax. The values for Pennsylvania are: **Y** Levies the tax **N** Does not levy the tax All other states are null.
PAY*n*.MunicipalIncomeTax	2 per PAY	PAY municipality income tax. The values for Pennsylvania are: **R** Resident **N** Non-resident **B** Both **X** None All other states are null.
PAY*n*.SchoolDistrictEMSTax	2 per PAY	PAY school district emergency municipal services tax. The Values for Pennsylvania are: **Y** Levies the tax

columnName	Max. Field Length (bytes)	Description
		N Does not levy the tax All other states are null.
PAY*n*.SchoolDistrictIncomeTax	2 per PAY	PAY school district income tax. The values for Pennsylvania are: **R** Resident **N** Non-resident **B** Both **X** N The values for Ohio are: **R** Resident **X** None All other states are null.

Property Tax District

The following table lists the output fields that contain Property Tax District (PTD) data. For more information on property tax districts, see **AssignGeoTAXInfo** on page 358. To include this data in the output, set **GeoTAXOutputRecordType** = P.

Note: AssignGeoTAXInfo returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 128: Property Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberPTDsFound	3	Number of PTDs returned.
PTD*n*.BoundaryBuffer.BufferRelation	2 per PTD	Indicates where in the district the address resides in relation to the edge of the district. One of the following: **P** The address is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The address is inside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**.

columnName	Max. Field Length (bytes)	Description
		B The address is outside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
PTD*n*.BoundaryBuffer.DistanceToBorder	10 per PTD	Indicates the distance from the address to the border of the district. The distance is in the units specified by the option DistanceUnits.
PTD*n*.BoundaryConfidence	4 per PTD	Provides the percentage overlap of the geocode to the PTD boundary polygon layer. The returned percentage value describes the probability that the point falls in the PTD boundary area.
PTD*n*.DistrictID	11 per PTD	PTD district ID.
PTD*n*.DistrictName	61 per PTD	PTD district name.
PTD*n*.DistrictType	2 per PTD	PTD district type. Only returned for Indiana. **R** Reporting district **B** Billing district
PTD*n*.JurisdictionID	11 per PTD	PTD jurisdiction ID.
PTD*n*.UpdateDate	7 per PTD	PTD update date.

Special Purpose Tax District

The following table lists the output fields that contain Special Purpose Tax District (SPD) data. For more information on special purpose tax districts, see **AssignGeoTAXInfo** on page 358. To include this data in the output, set **GeoTAXOutputRecordType** = S.

Note: AssignGeoTAXInfo returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 129: Special Purpose Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberSPDsFound	3	Number of SPDs returned.
SPD*n*.BoundaryBuffer.BufferRelation	2 per SPD	Indicates where in the district the address resides in relation to the edge of the district. One of the following:

columnName	Max. Field Length (bytes)	Description
		P The address is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The address is inside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The address is outside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
SPD*n*.BoundaryBuffer.DistanceToBorder	10 per SPD	Indicates the distance from the address to the border of the district. The distance is in the units specified by the option DistanceUnits.
SPD*n*.BoundaryConfidence	4 per SPD	Provides the percentage overlap of the geocode to the SPD boundary polygon layer. The returned percentage value describes the probability that the point falls in the SPD boundary area.
SPD*n*.CompiledDate	7 per SPD	SPD compiled date.
SPD*n*.DistrictCode	4 per SPD	3-digit district type code.
SPD*n*.DistrictName	61 per SPD	SPD name.
SPD*n*.DistrictNumber	6 per SPD	SPD district number.
SPD*n*.EffectiveDate	7 per SPD	SPD effective date.
SPD*n*.UpdateDate	7 per SPD	SPD update date.
SPD*n*.VersionDate	7 per SPD	SPD version date.

Sales and Use Tax Rates

The following table lists the output fields that contain the sales and use tax rate data.

To include tax rate data in the output, set **GeoTAXOutputRecordType** = B.

To select the tax rate type, set **TaxRate** to one of the following:

N	Do not return sales and use tax rates. (default)
G	Return the General sales and use tax rates.
A	Return the Automotive sales and use tax rates.
C	Return the Construction sales and use tax rates.
M	Return the Medical sales and use tax rates.

Note: You must be a licensed user of the Pitney Bowes Software Sales and Use Tax Rate file to use this feature.

Table 130: Sales and Use Tax Rate Output Fields

columnName	Max. Field Length (bytes)	Description
TaxRate.RC	2	Tax Rate return code denoting the level of match obtained against the Pitney Bowes Software Sales and Use Tax Rate file: **E** Exact match, using all 5 fields **P** Partial match, using 4 fields **A** Alternate match, using 3 fields **N** Record is default-coded based on valid state code. **Blank** No matching PB Software Sales and Use Tax Rate record found.
Municipal.SalesTaxRate	11	Municipality sales tax rate for the selected tax rate type.
County.SalesTaxRate	11	County sales tax rate for the selected tax rate type.
State.SalesTaxRate	11	State sales tax rate for the selected tax rate type.
SPD*n*.SalesTaxRate	11 per SPD	Sales tax rate for up to 10 Special Purpose Districts (SPD).
TaxRate.SalesTotal	11	The sum of the individual Municipal, County, State and SPD sales tax rates.
Municipal.UseTaxRate	11	Municipality use tax rate for the selected tax rate type.
County.UseTaxRate	11	County use tax rate for the selected tax rate type.
State.UseTaxRate	11	State use tax rate for the selected tax rate type.
SPD*n*.UseTaxRate	11 per SPD	Use tax rate for up to 10 Special Purpose Districts (SPD).
TaxRate.UseTotal	11	The sum of the individual Municipal, County, State and SPD use tax rates.

Error Reporting

The following table defines the error reporting output fields.

Table 131: Error Output Fields

columnName	Max. Field Length (bytes)	Description
GTX.ErrorCode	3	This field contains a return code if the GeoTAX engine experiences an abnormal termination. **Note:** This field contains the same set of codes returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform. The first character indicates the file (or set of files affected). **Blank** Matcher terminated normally **A** User Auxiliary file problem **CE** coubsub.txb file problem **CI** Confidence engine problem **D** Boundary file **F** User-defined boundary file problem **G** Address Matching engine problem **L** Licensing problem **S** State file problem **U** GeoTAX Auxiliary file problem **X** Combination of Street and state file problem **Z** zip.gsb file problem The second position is one of the following: **E** Fatal issue, program terminating **F** Expired database **I** Informational
GTX.ErrorDescription	81	If the GeoTAX engine experiences an abnormal termination, this field contains a text description of the reason. It is blank if GeoTAX terminated normally. **Note:** This field contains the same set of descriptions returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform. SI-"TS158 FILES NOT FOUND" SI-"TS158 FILES VINTAGE OR INCOMPLETE DB ERROR" SI-"STATE FILES NOT FOUND"

columnName	Max. Field Length (bytes)	Description
		SE-"STATE AND TS158 FILES NOT FOUND" SE-"STATE NOT FOUND AND TS158 VINTAGE ERROR" SI-"STATE FILES VINTAGE OR INCOMPLETE DB ERROR" SE-"STATE VINTAGE ERROR AND TS158 NOT FOUND" SE-"STATE AND TS158 FILES VINTAGE OR INCOMPLETE DB ERROR" GI-"STREET FILES NOT FOUND" XI-"STREET AND TS158 FILES NOT FOUND" XI-"STREET NOT FOUND AND TS158 FILES VINTAGE ERROR" XI-"STREET AND STATE FILES NOT FOUND" XE-"STREET STATE AND TS158 FILES NOT FOUND" XE-"STREET AND STATE NOT FOUND AND TS158 VINTAGE ERROR" XI-"STREET NOT FOUND AND STATE VINTAGE ERROR" XE-"STREET AND TS158 NOT FOUND AND STATE VINTAGE ERROR" XE-"STREET NOT FOUND AND STATE AND TS158 VINTAGE ERROR" GI-"STREET FILES VINTAGE OR INCOMPLETE DB ERROR" XI-"STREET VINTAGE ERROR AND TS158 NOT FOUND" XI-"STREET AND TS158 FILES VINTAGE OR INCOMPLETE DB ERROR" XI-"STREET VINTAGE ERROR AND STATE NOT FOUND" XE-"STREET VINTAGE ERROR AND STATE AND TS158 NOT FOUND" XE-"STREET AND TS158 VINTAGE ERROR AND STATE NOT FOUND" XI-"STREET AND STATE FILES VINTAGE OR INCOMPLETE DB ERROR" XE-"STREET AND STATE VINTAGE ERROR AND TS158 NOT FOUND" XE-"STREET STATE AND TS158 VINTAGE ERROR" LF-"INVALID FUNCTION PASSED TO GTDBLIO : " AI-"GENIO ERROR: FILE = G1GTAUX , FUNC = , ST = " UI-"GENIO ERROR: FILE = G1GTAX2 , FUNC = , ST = " XF-"The (DB Vintage) database has expired!" XF-"The (SPD file Vintage) SPD File has expired!" DI- "UNABLE TO VALIDATE BOUNDARY LICENSE" DI- "UNABLE TO OPEN BOUNDARY FILE" DI- "BOUNDARY FILE NOT FOUND" FI- "UNABLE TO VALIDATE USER BOUNDARY LICENSE"

columnName	Max. Field Length (bytes)	Description
		FI- "UNABLE TO OPEN USER BND FILE" FI- "USER BND FILE NOT FOUND"
GTX.WarnCode	3	This field contains warning codes returned by the GeoTAX engine. It is blank if no warnings were issued. A value of WN indicates a database will expire next month. **Note:** This field contains the same set of codes returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform.
GTX.WarnDescription	81	A text description of any warnings returned by the GeoTAX engine. **Note:** This field contains the same set of descriptions returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform.
Status	2	Reports the success or failure of the match attempt. **null** Success **F** Failure. Some examples of failures are your license expired or you did not select any output record types and fields for AssignGeoTAXInfo to return.
Status.Code	12	If AssignGeoTAXInfo could not process the address, this field will show the reason. Currently there is one possible value for this field: Invalid Address.
Status.Description	64	If AssignGeoTAXInfo could not process the address, this field will show a description of the failure. One of the following: TS158 FILES NOT FOUND TS158 FILES VINTAGE OR INCOMPLETE DB ERROR STATE FILES NOT FOUND STATE AND TS158 FILES NOT FOUND STATE NOT FOUND AND TS158 VINTAGE ERROR STATE FILES VINTAGE OR INCOMPLETE DB ERROR STATE VINTAGE ERROR AND TS158 NOT FOUND STATE AND TS158 FILES VINTAGE OR INCOMPLETE DB ERROR STREET FILES NOT FOUND STREET AND TS158 FILES NOT FOUND STREET NOT FOUND AND TS158 FILES VINTAGE ERROR STREET AND STATE FILES NOT FOUND STREET STATE AND TS158 FILES NOT FOUND

columnName	Max. Field Length (bytes)	Description
		STREET AND STATE NOT FOUND AND TS158 VINTAGE ERROR STREET NOT FOUND AND STATE VINTAGE ERROR STREET AND TS158 NOT FOUND AND STATE VINTAGE ERROR STREET NOT FOUND AND STATE AND TS158 VINTAGE ERROR STREET FILES VINTAGE OR INCOMPLETE DB ERROR STREET VINTAGE ERROR AND TS158 NOT FOUND STREET AND TS158 FILES VINTAGE OR INCOMPLETE DB ERROR STREET VINTAGE ERROR AND STATE NOT FOUND STREET VINTAGE ERROR AND STATE AND TS158 NOT FOUND STREET AND TS158 VINTAGE ERROR AND STATE NOT FOUND STREET AND STATE FILES VINTAGE OR INCOMPLETE DB ERROR STREET AND STATE VINTAGE ERROR AND TS158 NOT FOUND STREET STATE AND TS158 VINTAGE ERROR INVALID FUNCTION PASSED TO GTDBLIO : GENIO ERROR: FILE = G1GTAUX , FUNC = , ST = GENIO ERROR: FILE = G1GTAX2 , FUNC = , ST = The (DB Vintage) database has expired! The (SPD file Vintage) SPD File has expired! UNABLE TO VALIDATE BOUNDARY LICENSE UNABLE TO OPEN BOUNDARY FILE BOUNDARY FILE NOT FOUND UNABLE TO VALIDATE USER BOUNDARY LICENSE UNABLE TO OPEN USER BND FILE USER BND FILE NOT FOUND

CalculateDistance

CalculateDistance takes two sets of latitude/longitude coordinates as input, calculates the distance between the coordinates, and returns the distance between the two points.

CalculateDistance is part of the Enterprise Tax Module.

Input

CalculateDistance takes latitude and longitude information as input.

Note: Specify input using the DataTable class. For information on the DataTable class, see the "API Fundamentals" section.

Table 132: CalculateDistance Input Data

columnName	Description
FirstLatitude	Latitude of the first point for which you want distance returned.
FirstLatitude.Directional	First latitude directional. **N** North **S** South
FirstLongitude	Longitude of the first point for which you want distance returned.
FirstLongitude.Directional	First longitude directional. **E** East **W** West
SecondLatitude	Latitude of the second point for which you want distance returned.
SecondLatitude.Directional	Second latitude directional. **N** North **S** South
SecondLongitude	Longitude of the second point for which you want distance returned.
SecondLongitude.Directional	Second longitude directional. **E** East **W** West

Options

Table 133: Output Data and Format Options

optionName	Description
LatLongFormat	Indicates the format of the input latitude/longitude. The options are: **DegMinSec** For example 90 00 00N180 00 00W. **PreZero** (090000000N180000000W). Default. **PreZeroDecimal** (090.000000N180.000000W) **Decimal** (90.000000-180.000000) **DecimalAssumed** (90000000-180000000)
ReturnUnits	Indicates the measurement units returned for distance calculation: • Miles • Km • Feet • Meters

Output

CalculateDistance always returns the Confidence field to indicate the confidence in the output provided.

If CalculateDistance fails to process the data, it returns the fields Status, Status.Code, and Status.Descriptions. These fields provide information on why CalculateDistance failed to process the data. Some examples of failures are your license expired or you did not select any output record types and fields for CalculateDistance to return. The following table provides the record-level qualifiers and data outputs for CalculateDistance.

Table 134: CalculateDistance Output Fields

columnName	Max. Field Length (bytes)	Description
Distance	9	Distance between the two input coordinates in the units of measurement that you specified.
Status	2	Reports the success or failure of the match attempt: **null** Success **F** Failure
Status.Code	2	Reason for failure or error. If Status = F, Status.Code = Failure.
Status.Description	64	Description of the problem. If Status = F, Status.Description = Unable to compute distance.

ReverseGeoTAXInfoLookup

ReverseGeoTAXInfoLookup allows latitude/longitude coordinates to be supplied as input and identifies the tax districts that apply to the given coordinate. Specifically, ReverseGeoTAXInfoLookup can return the following information about a location:

- FIPS state codes and county codes
- State and county names
- MCD codes and names
- Place codes and names
- Boundary file districts
- Cross-reference tax keys
- The relationship of the input coordinates to user-defined polygons
- Sales and use tax rates, if licensed for the Pitney Bowes Software Sales and Use Tax Rate File

ReverseGeoTAXInfoLookup optionally includes enhanced tax jurisdiction information for a location, including:

- **Insurance premium districts**—Areas designated for the collection of taxes imposed on insurance policy premiums, based on the policy holder's address. Insurance premium districts are created by state governments.
- **Payroll tax districts**—Areas designated for the collection of taxes imposed on employers to support state or local government facilities and services, based on the employee's and/or employer's address. Examples include taxes collected for districts to pay for schools, police, or other services. Payroll tax districts are created by state or local governments.
- **Payroll system tax codes**—Codes that represent specific jurisdictions that collect payroll tax. Using payroll system tax codes has advantages over using the payroll tax district information returned by ReverseGeoTAXInfoLookup:

  - ReverseGeoTAXInfoLookup uses an additional database to determine payroll tax codes, resulting in more accurate payroll tax determination.
  - Many payroll systems use specific codes to determine withholding amounts. Since you can customize the payroll tax codes returned by ReverseGeoTAXInfoLookup, you can set up a process where ReverseGeoTAXInfo Lookup returns the exact payroll tax codes required by your payroll system, instead of returning jurisdictional IDs that must then be translated into the codes used by your system.

- **Property tax districts**—Areas designated for the collection of taxes imposed on property owners to support local government facilities and services, based on the property's location. Examples include taxes collected for districts to pay for schools, police, or other services. Property tax districts are created by local governments.
- **Special purpose tax districts**—Areas designated for the collection of taxes imposed on residents to support specialized services for residents of the district, based on the resident's address. Examples include services such as sewer service, transit service, or water resources. Special purpose tax districts are created by legislative action, court action, or public referendums. This optional information requires the use of boundary files which require an additional license. Contact your Pitney Bowes Software sales representative for more information.

Using the optional Pitney Bowes Software Sales and Use Tax Rate file, ReverseGeoTAXInfoLookup includes tax rate data for a location, including:

Tax rate type:

- General
- Automotive
- Medical
- Construction

Sales and/or use tax rates for:

- State
- County
- Municipality
- Up to 10 SPDs
- Total Rate - the sum of the individual state, county, municipality and SPD rates.

Required input format

The required format for the input coordinates is as follows:

Table 135: Required Input Latitude/Longitude Format

columnName	Format
InputLatitude	00.000000 or without the decimal point 00000000
InputLongitude	000.000000 or without the decimal point 000000000, or 00.000000 or without the decimal point 00000000

ReverseGeoTAXInfoLookup is part of the Enterprise Tax Module.

Geocoding Options

Reverse geocoding information lookup is the process of taking an input latitude/longitude coordinate and returning jurisdictional tax information. The geocoding options define the database to use for matching and specify the distance units and buffer distance to use when matching to a boundary file.

Table 136: ReverseGeoTAXInfoLookup Geocoding Options

optionName	Description
Database.GTX	Select the database resource to use in the reverse geocoding lookup process.
Boundary matching: The following options can be set when matching to a boundary file such as SPD, IPD, PAY, Place and MCD, or user-defined.	
DistanceUnits	Specifies the units in which to measure distance. One of the following: **Feet** Distances are measured in feet. (Default) **Km** Distances are measured in kilometers. **Meters** Distances are measured in meters. **Miles** Distances are measured in miles.
DefaultBufferWidth	Specifies the buffer width to use for tax district boundary files. The tax district boundary files are the Special Purpose District (SPD) file, the Insurance Premium District (IPD) file, the Payroll Tax District (PAY) file, the Personal Property Tax District (PTD) file, Place and MCD files. The default buffer width that you specify here can be overridden on a record-by-record basis using the BufferWidth input field. For more information on buffers, see **Buffering** on page 360.
DefaultUserBufferWidth	Specifies the buffer width to use for user-defined boundary files. Specify the distance in the units of measurement specified in the **Distance units** option. For information on buffers, see **Buffering** on page 360. The default buffer width that you specify here can be overridden on a record-by-record basis using the BufferWidth input field. **Note:** To use buffers, the user-defined boundary file must support buffers.

Output Data Options

Data options control the data returned by ReverseGeoTAXInfoLookup. **Table 137: ReverseGeoTAXInfo Lookup Data Options** on page 401 lists the output data options.

Table 137: ReverseGeoTAXInfo Lookup Data Options

optionName	Description
GeoTAXOutputRecordType	Select one or more of the following to obtain the type of data you want returned. ReverseGeoTAXInfo Lookup groups the output fields into record types. If you do not want all of the fields in a record type returned, do not select the check box, and list only those fields you want returned in Extra Output Fields. • **C**—Census • **T**—Tax Jurisdiction • **U**—User-defined boundary file • **W**—Payroll System Tax Codes

optionName	Description
	• **X**—Auxiliary File • **B**—PB Software Sales and Use Tax Rate file You can also specify one, and only one, of the following: **I** Insurance Premium Tax District (IPD) **P** Property Tax District (PTD) **R** Payroll Tax District (PAY) **S** Special Purpose Tax District (SPD) For a description of the fields in each output group, see **Output** on page 403. **Note:** If you specify `W`, also specify `R` to obtain the best payroll system tax code match possible.
TaxKey	If you integrate ReverseGeoTAXInfo Lookup with third-party tax compliance software from Vertex or Taxware, select which vendor you use. This controls the value returned in the GeoTAXKey output field. One of the following: **T** Return the Taxware jurisdiction code for the address. **Y** Return the Vertex jurisdiction code for the address. Select this option if you obtained a MatchMaster file from Vertex. **V** Return the Vertex jurisdiction code for the address. Select this option if you obtained a Vertex file from Pitney Bowes Software. **N** Do not return either the Taxware or Vertex jurisdiction codes (default).
TaxRate	Select the desired Sales tax rate type or None: **N** Do not return sales tax rates. (default) **G** Return the General sales tax rates. **A** Return the Automotive sales tax rates. **C** Return the Construction sales tax rates. Return the Medical sales tax rates.
OutputFields	Indicates the individual output fields you want returned. You can use this field instead of the Output Record Type to limit the output to those fields that are important to your current data needs. For a list of the fields included in each data type, see **Output** on page 403.

Output Format

Output format options control how ReverseGeoTAXInfo Lookup formats output data. The following table lists the output format options.

Table 138: ReverseGeoTAXInfoLookup Output Format Options

optionName	Description
OutputCasing	Specifies the casing of these output fields: County.Name, MCD.Name, Place.Name, IPDn.DistrictName, PAYn.DistrictName, SPDn.DistrictName, and PTCn.PayrollDescription. One of the following: **M** The output in mixed case (default). For example: Boulder. **U** The output in upper case. For example: BOULDER.

Output

Auxiliary File

The following table lists the output fields that contain auxiliary file data. To include auxiliary file data in the output, set **GeoTAXOutputRecordType** = X. The following table lists the output fields that contain tax jurisdiction data.

Table 139: Auxiliary File Output Fields

columnName	Max. Field Length (bytes)	Description
AuxiliaryData.AuxiliaryFile	301	Data retrieved as a result of an auxiliary match from the user-defined area of the auxiliary file.
AuxiliaryData.StateFile	201	Data retrieved as a result of a state match. Data content and format vary depending on the state file used.

Census

The census output fields contains census information from the U.S. Census, including Minor Civil Divisions (MCDs) and Census County Division (CCD) names and codes. MCDs are the primary political or administrative divisions of a county, representing many kinds of legal entities with a variety of governmental and administrative functions. CCDs are established in states where there are no legally established MCDs. The Census Bureau recognizes MCDs in 28 states and has established CCDs in 21 states. The District of Columbia has no primary divisions, and the city of Washington, DC is considered equivalent to an MCD for data presentation purposes.

Census data also contains the Federal Information Processing Standards (FIPS) codes for each state and county. The FIPS State Code and the FIPS County Code are both used by the Census Bureau to identify these geographic units.

The following table lists the output fields that contain census data. To include census data in the output, set **GeoTAXOutputRecordType** = C.

Table 140: Census Data Output Fields

columnName	Max. Field Length (bytes)	Description
County.Code	4	Three-digit Federal Information Processing Standards (FIPS) county code extracted from the Census.BlockCode. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.
County.Name	26	Name of the county. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.
MCD.Code	6	Minor Civil Division/Census County Division (MCD/CCD) Code.
MCD.Name	41	Minor Civil Division/Census County Division (MCD/CCD) name.
MCD.PointStatus	2	An address can be compared to a Minor Civil Division (MCD)/county subdivision file (Cousub.txt). This output field returns the result for a comparison between the matched geocode location to the polygon defined by the Cousub.txb file. For more information on buffers, see **Buffering** on page 360. **Note:** The buffer distance for Cousub.txb is internally set to zero and cannot be modified. **P** The point is in the polygon. **I** The point is in the buffer area inside the polygon. **B** The point is in the buffer area and outside of the polygon. **blank** Polygon not found.
MCD.DistanceToBorder	10	Returns the distance in feet between the matched address point to the polygon defined by the Cousub.txb file.
StateCode	3	Two-digit Federal Information Processing Standards (FIPS) state code extracted from the Census.BlockCode. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.

Input Latitude/Longitude

ReverseGeoTAXInfoLookup always returns the input coordinates as part of the output. The input latitude/longitude fields are returned as input from the data. ReverseGeoTAXInfoLookup does not change these input values.

Table 141: Input Latitude/Longitude Output Fields

columnName	Max. Field Length (bytes)	Description
InputLatitude	12	Input latitude.
InputLongitude	12	Input longitude.

Payroll System Tax Code

The following table lists the output fields that contain Payroll System Tax Code (PTC) data. For more information on payroll tax districts, see **Reverse GeoTAX Info Lookup**. To include this data in the output, set **GeoTAXOutputRecordType** = W.

Note: ReverseGeoTAXInfoLookup returns up to six payroll tax codes per input location.

Table 142: Payroll System Tax Code Output Fields

columnName	Max. Field Length (bytes)	Description
NumberPTCsFound	2	The number of payroll system tax codes found for this location.
PTC*n*.MatchCode	2 per PTC	Indicates the level of match obtained for the location. In order from most specific match to least, the possible match codes are: **P** The address was matched to a specific Payroll District ID. This is the most specific match. **G** The address was matched to a GNIS Code. **F** The address was matched to a county's FIPS code. **S** The address was matched to a state's FIPS code. This is the least specific match.
PTC*n*.PayrollCode	16 per PTC	A code that represents a taxing authority in a payroll application. This is a user-defined code. The specific codes are determined by the payroll application that utilizes the data returned by ReverseGeoTAXInfo Lookup.
PTC*n*.PayrollDescription	41 per PTC	A description of the purpose of this payroll code.

columnName	Max. Field Length (bytes)	Description
PTC*n*.PayrollFlag	7 per PTC	A user-defined flag from the PTC database.
StateCounty	33	The state abbreviation and county name.

Tax Jurisdiction

Tax jurisdiction data contains information about the "place" for the location. A "place" is a geographic area defined on the basis of population criteria that vary by state; or, an area recognized as significant because it is located in an incorporated municipality. Places are used to determine tax jurisdiction.

The following table lists the output fields that contain tax jurisdiction data. To include tax jurisdiction data in the output, set **GeoTAXOutputRecordType** = T.

Table 143: Tax Jurisdiction Output Fields

columnName	Max. Field Length (bytes)	Description
GeoTAXKey	10	The value in this field varies depending on the option you specified in the TaxKey option: If you specified `T`, GeoTAXKey contains the proprietary codes used in Taxware tax compliance software. You can use this code in your Taxware application to find out the tax rate for the jurisdiction. If you specified `Y` or `V`, GeoTAXKey contains the proprietary Vertex® jurisdiction code (comprised of a two-digit Vertex® state code, three-digit FIPS county code, and four-digit Vertex® city code). You can use this code in your Vertex® application to find out the tax rate for the jurisdiction.
GeoTAXKey.MatchCode	2	Return code denoting the level of match obtained against the Vertex or Taxware cross reference files. **E** Exact match using five fields: FIPS state code, FIPS county code, FIPS or GNIS place code, ZIP Code, and FIPS place name. **P** Partial match using four fields: FIPS state code, FIPS county code, FIPS or GNIS place code, and ZIP Code. **A** Alternate match using two fields: ZIP Code, FIPS place name. **N** Record is default coded based on valid state code. **null** No matching record found.

columnName	Max. Field Length (bytes)	Description
GeoTAXKey.MatchLevel	12	A description of the value returned in the GeoTAXKey.MatchCode field. **Exact** Exact match. See description in GeoTAXKey.MatchCode. **Partial** Partial match. See description in GeoTAXKey.MatchCode. **Alternate** Alternate match. See description in GeoTAXKey.MatchCode. **DefaultCode** Record is default coded. See description in GeoTAXKey.MatchCode. **NoMatch** No matching record found.
GNISCode	10	Unique nine-digit Geographic Names Information System (GNIS) code. **Note:** This field is always included in the output regardless of whether or not you choose to include census data in the output.
Place.ClassCode	3	Place class code. Place class codes are used to determine the proper taxing jurisdictions **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.Code	6	An identifier for a specific place. A "place" is a geographic area defined on the basis of population criteria that vary by state. Or, an area recognized as significant because it is located in an incorporated municipality. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.IncorporatedFlag	8	Indicates whether the address is located in an incorporated or unincorporated place. A "place" is a geographic area defined on the basis of population criteria that vary by state. Or, an area recognized as significant because it is located in an incorporated municipality. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output. **Inc** Incorporated place code. **Uninc** Unincorporated place code.

columnName	Max. Field Length (bytes)	Description
		Unknown Incorporation status unknown.
Place.LastAnnexedDate	8	Last annexed date, in the format MM/YYYY, representing the month and year of the most recent boundary change or the most recent available boundary information. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.LastUpdatedDate	8	Last updated date, in the format MM/YYYY, reflecting the month and year when TomTom updated the database to reflect attribute (name change, FIPS change, etc.) or boundary edits to the Place. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.LastVerifiedDate	8	Last verified date, in the format MM/YYYY, representing the month and year that TomTom verified municipality change information. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.Name	41	The name of the "place" for the location. A "place" is a geographic area defined on the basis of population criteria that vary by state. Or, an area recognized as significant because it is located in an incorporated municipality. **Note:** This field is always included in the output regardless of whether or not you choose to include tax jurisdiction data in the output.
Place.PointStatus	2	Returns the result for a comparison between the matched geocode location to the polygon defined by the Place.txb file. For more information on buffers, see **Buffering** on page 360. **P** The point is in the polygon. **I** The point is in the buffer area inside the polygon. **B** The point is in the buffer area and outside of the polygon. **blank** Polygon not found.

columnName	Max. Field Length (bytes)	Description
Place.DistanceToBorder	10	Returns the distance in feet between the matched address point to the polygon defined by the Place.txb file.

User-Defined Boundary File

The following table lists the output fields that contain data returned from user-defined boundary files. To include this data in the output, set **GeoTAXOutputRecordType** = U.

Note: ReverseGeoTAXInfoLookup can return up to 10 user-defined areas for each input location.

Table 144: Output Fields for User-Defined Boundary Files

columnName	Max. Field Length (bytes)	Description
NumberUserBoundariesFound	3	The number of user-defined polygons found for the address.
UserBoundary*n*.BoundaryDescription	51 per User Boundary	A description of the polygon.
UserBoundary*n*.BoundaryID	11 per User Boundary	The ID of the polygon as specified in the user-defined boundary file.
UserBoundary*n*.BufferRelation	2 per User Boundary	Indicates where in the polygon the location resides in relation to the edge of the area. One of the following: **P** The geocode is inside the polygon at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The geocode is inside the polygon but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The geocode is outside the polygon but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.

columnName	Max. Field Length (bytes)	Description
UserBoundary*n*.DistanceToBorder	10 per User Boundary	Indicates the distance in feet from the input location to the border of the polygon.
UserBoundary*n*.SupplementalBoundaryID	11 per User Boundary	A supplemental ID as specified in the user-defined boundary file.

Insurance Premium Tax Districts

The following table lists the output fields that contain Insurance Premium Tax Districts (IPD) data. For more information on insurance premium tax districts, see **ReverseGeoTAXInfoLookup** on page 399. To include IPD data in the output, set **GeoTAXOutputRecordType** = I.

Note: ReverseGeoTAXInfoLookup returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 145: Insurance Premium Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberIPDsFound	3	The number of Insurance Premium Tax Districts found for the location.
IPD*n*.BoundaryBuffer.BufferRelation	2 per IPD	Indicates where in the district the location resides in relation to the edge of the district. One of the following: **P** The location is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The location is inside the district but is close to the edge. This indicates that the location is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The location is outside the district but is close to the edge. This indicates that the location is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
IPD*n*.BoundaryBuffer.DistanceToBorder	10 per IPD	Indicates the distance in feet from the location to the border of the district.

columnName	Max. Field Length (bytes)	Description
IPD*n*.DistrictID	11 per IPD	IPD ID.
IPD*n*.DistrictName	61 per IPD	IPD name.
IPD*n*.DistrictType	7 per IPD	IPD district type.
IPD*n*.UpdateDate	7 per IPD	IPD update date (MMYYYY).
IPD*n*.VersionDate	7 per IPD	IPD compiled date (MMYYYY).
IPD*n*.Notes	21 per IPD	Tax code descriptions. For example: 01, 33, A, B
IPD*n*.ChangeDate	7 per IPD	IPD change date.
IPD*n*.EffectiveDate	7 per IPD	MMDDYY - Identifies when district becomes active - State supplied For example: 010108
IPD*n*.ExpirationDate	7 per IPD	MMDDYY - Identifies when district becomes inactive - State supplied For example: 063009
IPD*n*.FireRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.FireFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semi colon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.CasualtyRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.CasualtyFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.VehicleRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.VehicleFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"

columnName	Max. Field Length (bytes)	Description
IPD*n*.MarineRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.MarineFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.HealthRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.HealthFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.LifeRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.LifeFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.OtherRate	21 per IPD	Format is dependent on associated flag Possible Values: .13, 15.00 or 3;7
IPD*n*.OtherFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"
IPD*n*.MinimumRate	21 per IPD	Format is dependent on associated flag For example: .13, 15.00 or 3;7
IPD*n*.MinimumFlag	6 per IPD	P - Percentage; .1 = 10%, .0575 = 5.75% F - Flat Fee dollar amount M - Multiple Percentages has a semicolon as a delimiter. 3;7 = "3% or 7%"

Payroll Tax Districts

The following table lists the output fields that contain Payroll Tax District (PAY) data. For more information on payroll tax districts, see **Reverse GeoTAX Info Lookup**. To include this data in the output, set **GeoTAXOutputRecordType** = R.

Note: ReverseGeoTAXInfoLookup returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 146: Payroll Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberPAYsFound	3	Number of payroll tax districts found for the location.
PAY*n*.BoundaryBuffer.BufferRelation	2 per PAY	Indicates where in the district the location resides in relation to the edge of the district. One of the following: **P** The location is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The location is inside the district but is close to the edge. This indicates that the location is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The location is outside the district but is close to the edge. This indicates that the location is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
PAY*n*.BoundaryBuffer.DistanceToBorder	10 per PAY	Indicates the distance in feet from the location to the border of the district.
PAY*n*.DistrictID	11 per PAY	PAY district ID.
PAY*n*.DistrictName	61 per PAY	PAY district name.
PAY*n*.DistrictType	7 per PAY	PAY district type.
PAY*n*.ID	11 per PAY	PAY ID.
PAY*n*.MunicipalEMSTax	2 per PAY	PAY municipality emergency municipal services tax. The values for Pennsylvania are: **Y** Levies the tax **N** Does not levy the tax All other states are null.
PAY*n*.MunicipalIncomeTax	2 per PAY	PAY municipality income tax. The values for Pennsylvania are: **R** Resident **N** Non-resident

columnName	Max. Field Length (bytes)	Description
		B Both **X** None All other states are null.
PAY*n*.SchoolDistrictEMSTax	2 per PAY	PAY school district emergency municipal services tax. The Values for Pennsylvania are: **Y** Levies the tax **N** Does not levy the tax All other states are null.
PAY*n*.SchoolDistrictIncomeTax	2 per PAY	PAY school district income tax. The values for Pennsylvania are: **R** Resident **N** Non-resident **B** Both **X** N The values for Ohio are: **R** Resident **X** None All other states are null.

Property Tax District

The following table lists the output fields that contain Property Tax District (PTD) data. For more information on property tax districts, see **ReverseGeoTAXInfoLookup** on page 399. To include this data in the output, set **GeoTAXOutputRecordType** = P.

Note: ReverseGeoTAXInfoLookup returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 147: Property Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberPTDsFound	3	The number of Property Tax Districts found for the location.
PTD*n*.BoundaryBuffer.BufferRelation	2 per PTD	Indicates where in the district the location resides in relation to the edge of the district. One of the following:

columnName	Max. Field Length (bytes)	Description
		P The location is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The location is inside the district but is close to the edge. This indicates that the location is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The location is outside the district but is close to the edge. This indicates that the address is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
PTD*n*.BoundaryBuffer.DistanceToBorder	10 per PTD	Indicates the distance in feet from the location to the border of the district.
PTD*n*.DistrictID	11 per PTD	PTD district ID.
PTD*n*.DistrictName	61 per PTD	PTD district name.
PTD*n*.DistrictType	2 per PTD	PTD district type. Only returned for Indiana. **R** Reporting district **B** Billing district
PTD*n*.JurisdictionID	11 per PTD	PTD jurisdiction ID.
PTD*n*.UpdateDate	7 per PTD	PTD update date.

Special Purpose Tax Districts

The following table lists the output fields that contain Special Purpose Tax Districts (SPD) data. For more information on special purpose tax districts, see **Reverse GeoTAX Info Lookup**. To include this data in the output, set **GeoTAXOutputRecordType** = S.

Note: ReverseGeoTAXInfoLookup returns multiple districts for IPDs, SPDs, PTDs, and PAYs.

Table 148: Special Purpose Tax District Output Fields

columnName	Max. Field Length (bytes)	Description
NumberSPDsFound	3	Number of Special Purpose Tax Districts found for the location.
SPD*n*.BoundaryBuffer.BufferRelation	2 per SPD	Indicates where in the district the location resides in relation to the edge of the district. One of the following: **P** The location is inside the district at a distance from the edge that is greater than the specified buffer width. Buffer width is specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **I** The location is inside the district but is close to the edge. This indicates that the location is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. **B** The location is outside the district but is close to the edge. This indicates that the location is in the buffer area specified either by the option DefaultBufferWidth or by the input field **BufferWidth**. For more information, see **Buffering** on page 360.
SPD*n*.BoundaryBuffer.DistanceToBorder	10 per SPD	Indicates the distance in feet from the address to the border of the district.
SPD*n*.CompiledDate	7 per SPD	SPD compiled date.
SPD*n*.DistrictCode	4 per SPD	3-digit district type code.
SPD*n*.DistrictName	61 per SPD	SPD name.
SPD*n*.DistrictNumber	6 per SPD	SPD district number.
SPD*n*.EffectiveDate	7 per SPD	SPD effective date.
SPD*n*.UpdateDate	7 per SPD	SPD update date.
SPD*n*.VersionDate	7 per SPD	SPD version date.

Sales and Use Tax Rates

The following table lists the output fields that contain the sales and use tax rate data.

To include tax rate data in the output, set **GeoTAXOutputRecordType** = B.

To select the tax rate type, set **TaxRate** to one of the following:

N	Do not return sales and use tax rates. (default)
G	Return the General sales and use tax rates.
A	Return the Automotive sales and use tax rates.
C	Return the Construction sales and use tax rates.
M	Return the Medical sales and use tax rates.

Note: You must be a licensed user of the Pitney Bowes Software Sales and Use Tax Rate file to use this feature.

Table 149: Sales and Use Tax Rate Output Fields

columnName	Max. Field Length (bytes)	Description
TaxRate.RC	2	Tax Rate return code denoting the level of match obtained against the Pitney Bowes Software Sales and Use Tax Rate file: **E** Exact match, using all 5 fields **P** Partial match, using 4 fields **A** Alternate match, using 3 fields **N** Record is default-coded based on valid state code. **Blank** No matching PB Software Sales and Use Tax Rate record found.
Municipal.SalesTaxRate	11	Municipality sales tax rate for the selected tax rate type.
County.SalesTaxRate	11	County sales tax rate for the selected tax rate type.
State.SalesTaxRate	11	State sales tax rate for the selected tax rate type.
SPD*n*.SalesTaxRate	11 per SPD	Sales tax rate for up to 10 Special Purpose Districts (SPD).
TaxRate.SalesTotal	11	The sum of the individual Municipal, County, State and SPD sales tax rates.
Municipal.UseTaxRate	11	Municipality use tax rate for the selected tax rate type.
County.UseTaxRate	11	County use tax rate for the selected tax rate type.
State.UseTaxRate	11	State use tax rate for the selected tax rate type.
SPD*n*.UseTaxRate	11 per SPD	Use tax rate for up to 10 Special Purpose Districts (SPD).
TaxRate.UseTotal	11	The sum of the individual Municipal, County, State and SPD use tax rates.

Error Reporting

The following table defines the error reporting output fields.

Table 150: Error Output Fields

columnName	Max. Field Length (bytes)	Description
GTX.ErrorCode	3	This field contains a return code if the GeoTAX engine experiences an abnormal termination. **Note:** This field contains the same set of codes returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform. The first character indicates the file (or set of files affected). **Blank** Matcher terminated normally **A** User Auxiliary file problem **CE** coubsub.txb file problem **CI** Confidence engine problem **D** Boundary file **F** User-defined boundary file problem **G** Address Matching engine problem **L** Licensing problem **S** State file problem **U** GeoTAX Auxiliary file problem **X** Combination of Street and state file problem **Z** zip.gsb file problem The second position is one of the following: **E** Fatal issue, program terminating **F** Expired database **I** Informational
GTX.ErrorDescription	81	If the GeoTAX engine experiences an abnormal termination, this field contains a text description of the reason. It is blank if GeoTAX terminated normally. The maximum length is 80. **Note:** This field contains the same set of descriptions returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform.

columnName	Max. Field Length (bytes)	Description
		SI-"TS158 FILES NOT FOUND" SI-"TS158 FILES VINTAGE OR INCOMPLETE DB ERROR" SI-"STATE FILES NOT FOUND" SE-"STATE AND TS158 FILES NOT FOUND" SE-"STATE NOT FOUND AND TS158 VINTAGE ERROR" SI-"STATE FILES VINTAGE OR INCOMPLETE DB ERROR" SE-"STATE VINTAGE ERROR AND TS158 NOT FOUND" SE-"STATE AND TS158 FILES VINTAGE OR INCOMPLETE DB ERROR" GI-"STREET FILES NOT FOUND" XI-"STREET AND TS158 FILES NOT FOUND" XI-"STREET NOT FOUND AND TS158 FILES VINTAGE ERROR" XI-"STREET AND STATE FILES NOT FOUND" XE-"STREET STATE AND TS158 FILES NOT FOUND" XE-"STREET AND STATE NOT FOUND AND TS158 VINTAGE ERROR" XI-"STREET NOT FOUND AND STATE VINTAGE ERROR" XE-"STREET AND TS158 NOT FOUND AND STATE VINTAGE ERROR" XE-"STREET NOT FOUND AND STATE AND TS158 VINTAGE ERROR" GI-"STREET FILES VINTAGE OR INCOMPLETE DB ERROR" XI-"STREET VINTAGE ERROR AND TS158 NOT FOUND" XI-"STREET AND TS158 FILES VINTAGE OR INCOMPLETE DB ERROR" XI-"STREET VINTAGE ERROR AND STATE NOT FOUND" XE-"STREET VINTAGE ERROR AND STATE AND TS158 NOT FOUND" XE-"STREET AND TS158 VINTAGE ERROR AND STATE NOT FOUND" XI-"STREET AND STATE FILES VINTAGE OR INCOMPLETE DB ERROR" XE-"STREET AND STATE VINTAGE ERROR AND TS158 NOT FOUND" XE-"STREET STATE AND TS158 VINTAGE ERROR" LF-"INVALID FUNCTION PASSED TO GTDBLIO : " AI-"GENIO ERROR: FILE = G1GTAUX , FUNC = , ST = " UI-"GENIO ERROR: FILE = G1GTAX2 , FUNC = , ST = " XF-"The (DB Vintage) database has expired!" XF-"The (SPD file Vintage) SPD File has expired!" DI- "UNABLE TO VALIDATE BOUNDARY LICENSE"

columnName	Max. Field Length (bytes)	Description
		DI- "UNABLE TO OPEN BOUNDARY FILE" DI- "BOUNDARY FILE NOT FOUND" FI- "UNABLE TO VALIDATE USER BOUNDARY LICENSE" FI- "UNABLE TO OPEN USER BND FILE" FI- "USER BND FILE NOT FOUND"
GTX.WarnCode	3	This field contains warning codes returned by the GeoTAX engine. It is blank if no warnings were issued. A value of `WN` indicates a database will expire next month. **Note:** This field contains the same set of codes returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform.
GTX.WarnDescription	81	A text description of any warnings returned by the GeoTAX engine. **Note:** This field contains the same set of descriptions returned by the standalone GeoTAX software and is intended for users who have migrated from GeoTAX to Spectrum™ Technology Platform.

Summary Reports

Assign GeoTAX Info Summary Report

A pre-configured report can be generated for a job. This is supported for a job run through either the Assign GeoTAX Info or the Reverse GeoTAX Info Lookup stage. The report summarizes general information about the job, such as the settings, number of records processed, performance statistics, and the database used. It also contains detailed statistics about the results of the job.

The summary report contains the following sections for Assign GeoTAX Info and Reverse GeoTAX Info Lookup except where noted.

Job Summary

This section contains information about the software and databases used for the job.

- **Software Version**—The version of the underlying software used by AssignGeoTAXInfo. Note that this is not the same as the Spectrum™ Technology Platform version number.
- **Input Mode**—The type of input data.
  - `Address Input`— Indicates the data was generated from a job using the AssignGeoTAXInfo stage.
  - `Latitude-Longitude Input`—Indicates the data was generated from a job using the ReverseGeoTAXInfo Lookup stage.
- **Street Database Vendor**—The street database vendor used for matching.

- **Street Database Version**—The version of the Street portion of the Master File database. For information on the Master File, see **Enterprise Tax Module** on page 354.
- **Cross Reference Database Version**—The version of the cross-reference database used in this job. Cross-reference databases are used to determine jurisdiction codes for use with third-party tax software. For information on the Cross Reference databases, see **Enterprise Tax Module** on page 354.

Address Matching Summary

Note: This section of the summary is not provided for jobs using the Reverse GeoTAX Info Lookup stage.

This section describes the input address match counts.

- **Total Match Attempts**—Address matches attempted in this job.
- **Total Records Matched**—Input addresses that were matched to known locations.
- **Total Unmatched Records**—Input addresses that could not be matched to known addresses.

Address Matching Levels

Note: This section of the summary is not provided for jobs using the Reverse GeoTAX Info Lookup stage.

This section lists the counts and percentages of the types of address matches performed by Assign GeoTAX Info.

- **User-Defined Auxiliary File Matches**—The number of address matches made to the user-defined auxiliary file. For information on a user-defined auxiliary file, see **Enterprise Tax Module** on page 354.
- **GeoTAX Auxiliary File Matches**—The number of address matches made to the GeoTAX Auxiliary file. For information on the GeoTAX Auxiliary file, see **Enterprise Tax Module** on page 354.
- **State File Matches**—The number of address matches made to a state-supplied files.
- **Landmark Auxiliary Matches**—The number of address matches made to the Landmark Auxiliary file.
- **Point-Level Matches**—The number of address matches made using point-level data.
- **Address-Level Matches**—The number of addresses matches made to an approximate location on a street segment.
- **Street Intersection-Level Matches**—The number of address matches made to a street intersection.
- **Street-Centroid Level Matches**—The number of address matches that were made along the matched street segment.
- **ZIP + 4-Level Matches**—The number of address matches made to an area defined by a ZIP + 4 Code.
- **ZIP-Level Matches**—The number of address matches made to an area defined by a 5-digit ZIP Code.
- **Fallback Geographic Matches**—The number of address matches that were made to a city or state centroid.

Latitude/Longitude Matching

Note: This section of the summary is not provided for jobs using the Reverse GeoTAX Info Lookup stage.

This section lists the counts and percentages of the types of geocodes determined by Assign GeoTAX Info. This section describes how precisely AssignGeoTAXInfo determined latitude/longitude coordinates of an address. These counts are based on match codes. For more information, on match codes, see **Latitude/Longitude** on page 377.

- **Total Lat/Long Matches**—Addresses that AssignGeoTAXInfo determined latitude/longitude coordinates. Addresses included in this count have any value other than null in the **LatLong.MatchCode** output field.

- **Total Lat/Long Unmatched**—Addresses that AssignGeoTAXInfo was unable to determine latitude/longitude. Addresses included in this count have a value of null in the **LatLong.MatchCode** output field.
- **GeoTAX Auxiliary-level Matches**—The count of geocodes determined using the GeoTAX Auxiliary File.
- **Landmark-Level Matches**—The count of geocodes determined using the Landmark Auxiliary File.
- **Point-level Matches**—The count of geocodes that represent the actual location of the address using point-level data. This is the most accurate type of geocode.
- **Address-Level Matches**—The count of geocodes made to the actual location of the address. Addresses included in this count have a value of R in the **LatLong.MatchCode** output field.
- **ZIP + 4-Level Matches**—The count of geocodes made that represent the center of the ZIP + 4 code in which the address is located. Addresses included in this count have a value of 4 in the **LatLong.MatchCode** output field.
- **Census Block Group-Level Matches**—The count of geocodes that represent the center of the address's Census block group. Addresses included in this count have a value of B in the **LatLong.MatchCode** output field.
- **ZIP + 2-Level Matches**—The count of geocodes that represent the center of the address's ZIP + 2 code. Addresses included in this count have a value of 2 in the **LatLong.MatchCode** output field.
- **Census Tract-Level Matches**—The count of geocodes that represent the center of the address's Census tract. Addresses included in this count have a value of T in the **LatLong.MatchCode** output field.
- **ZIP Code-Level Matches**—The count of geocodes that represent the center of the ZIP Code in which the address is located. Addresses included in this count have a value of Z or 5 in the **LatLong.MatchCode** output field.
- **City-Centroid Level Matches**—The count of geocodes that represent the center of a city.
- **State-Centroid Level Matches**—The count of geocodes that represent the center of a state.

Census Matching

This section describes how precisely AssignGeoTAXInfo determined an address location within the statistical areas defined by the U.S. Census.

- **Census Tracts Determined**—Addresses that AssignGeoTAXInfo determined a census tract. Addresses included in this count have a value in the **Census.Tract** output field.
- **Census Tracts Not Determined**—Addresses that AssignGeoTAXInfo was unable to determine a census tract. Addresses included in this count have no value in the **Census.Tract** output field.
- **State Codes Determined**—Addresses that AssignGeoTAXInfo determined a state. Addresses included in this count have a value in the **StateCode** output field.
- **State Codes Not Determined**—Addresses that AssignGeoTAXInfo did not determine a state. Addresses included in this count have no value in the **StateCode** output field.
- **County Codes Determined**—Addresses that AssignGeoTAXInfo determined a county. Addresses included in this count have a value in the **County.Code** output field.
- **County Codes Not Determined**—Addresses that AssignGeoTAXInfo did not determine a county. Addresses included in this count have no value in the **County.Code** output field.
- **Census Block Group Determined**—Addresses that AssignGeoTAXInfo determined a block group. Addresses included in this count have a value in the **Census.BlockCode** output field.
- **Census Block Groups Not Determined**—Addresses that AssignGeoTAXInfo was unable to determine a block group Addresses included in this count have no value in the **Census.BlockCode** output field.
- **CBSA Codes Determined**—Addresses that AssignGeoTAXInfo determined a core based statistical area (CBSA). Addresses included in this count have a value in the **CBSA.Code** output field.
- **CBSA Codes Not Determined**—Addresses that AssignGeoTAXInfo did not determine a core based statistical area (CBSA). Addresses included in this count have no value in the **CBSA.Code** output field.

- **MCD/CCD Codes Determined**—Addresses that AssignGeoTAXInfo determined a Minor Civil Division/Census County Division code. Addresses included in this count have a value in the **MCD.Code** output field.
- **MCD/CCD Codes Not Determined**—Addresses that AssignGeoTAXInfo was unable to determine a Minor Civil Division/Census County Division code. Addresses included in this count have no value in the **MCD.Code** output field.

Tax Jurisdiction Matching

This section describes how accurately AssignGeoTAXInfo determined an address place. The place information is used to determine tax jurisdictions.

- **Place Codes Determined**—Addresses that AssignGeoTAXInfo determined a place code. Addresses included in this count have a value in the **Place.Code** output field.
- **Place Codes Not Determined**—Addresses that AssignGeoTAXInfo was unable to determine a place code. Addresses included in this count have no value in the **Place.Code** output field.
- **Place Names Determined**—Addresses that AssignGeoTAXInfo determined a place name. Addresses included in this count have a value in the **Place.Name** output field.
- **Place Class Codes Determined**—Addresses that AssignGeoTAXInfo determined a place code. Addresses included in this count have a value in the **Place.ClassCode** output field.
  - **Incorporated Places**—Addresses that reside in an incorporated municipality. Addresses included in this count have a value of Inc in the **Place.IncorporatedFlag** output field.
  - **Unincorporated Places**—Addresses that reside in a place that is not an incorporated municipality. Addresses included in this count have a value of Uninc in the **Place.IncorporatedFlag** output field.
- **Place Last Annexed Date Found**—Places that AssignGeoTAXInfo determined the last date of annexation. Addresses included in this count have a value in the **Place.LastAnnexedDate** output field.
- **Place Last Verified Date Found**—Places that AssignGeoTAXInfo determined the last date that the place data was verified by the data provider. Addresses included in this count have a value in the **Place.LastVerifiedDate** output field.
- **Place Last Updated Date Found**—Places that AssignGeoTAXInfo determined the date that the place data was last updated by the data provider. Addresses included in this count have a value in the **Place.LastUpdatedDate** output field.

GeoTAX Key Matching

This section describes how accurately AssignGeoTAXInfo determined GeoTAX keys. A GeoTAX key is used in conjunction with software from a third party to determine tax rates. If you do not choose to return a GeoTAX key, this section is blank.

- **GeoTAX Cross Reference Match Attempts**—Addresses that AssignGeoTAXInfo attempted to match to an cross reference file. These files are used to determine codes for use with third party tax software.
- **Unsuccessful GeoTAX Cross Reference Matches**—Addresses that AssignGeoTAXInfo attempted to determine a code but was unable.
- **Successful GeoTAX Cross Reference Matches**—Addresses that AssignGeoTAXInfo determined a code using a cross reference file. Addresses included in this count have a value in the **GeoTAXKey** output field.
  - **Exact Matches Achieved**—Addresses matched exactly to a GeoTAX key. For more information about GeoTAX key match levels, see the description of the GeoTAXKey.MatchCode output field under **Tax Jurisdiction** on page 381.
  - **Partial Matches Achieved**—Addresses partially matched to a GeoTAX key. For more information about GeoTAX key match levels, see the description of the GeoTAXKey.MatchCode output field under **Tax Jurisdiction** on page 381.

- **Alternate Matches Achieved**—Addresses matched to a GeoTAX key using an alternate, less accurate, method. For more information about GeoTAX key match levels, see the description of the GeoTAXKey.MatchCode output field under **Tax Jurisdiction** on page 381.
- **Default Matches Achieved**—Addresses matched to a GeoTAX key only at the state level. For more information about GeoTAX key match levels, see the description of the GeoTAXKey.MatchCode output field under **Tax Jurisdiction** on page 381.

Tax District Matching

This section describes the number of matches to specific types of tax districts. The specific type of tax districts displayed in this section vary based the district type selected in the **Tax district** field.

Note: If you specify a buffer width in the **Tax district buffer** field, a single location could be counted in more than one district due to buffering. For more information about buffering, see **Buffering** on page 360.

- **Successful Payroll Tax File Matches**—Addresses that AssignGeoTAXInfo determined a payroll tax code using the Payroll Tax District boundary file. For more information on Payroll Tax Districts, see **Payroll Tax District** on page 388.
- **Unsuccessful Payroll Tax File Matches**—Addresses that AssignGeoTAXInfo was unable to determine a payroll tax code using the Payroll Tax District boundary file. For more information on Payroll Tax Districts, see **Payroll Tax District** on page 388.
- **Locations in Special Purpose Districts**—Addresses that reside in at least one Special Purpose District. For more information on Special Purpose Districts, see **Special Purpose Tax District** on page 391.
- **Locations in 1 Special Purpose District**—Addresses that reside in a single Special Purpose District. Addresses included in this count have a value of 1 in the **NumberSPDsFound** output field.
- **Locations in 2 Special Purpose Districts**—Addresses that reside in two overlapping Special Purpose Districts. Addresses included in this count have a value of 2 in the **NumberSPDsFound** output field.
- **Locations in 3 or 4 Special Purpose Districts**—Addresses that reside in three or four overlapping Special Purpose Districts. Addresses included in this count have a value of 3 or 4 in the **NumberSPDsFound** output field.
- **Locations in 5+ Special Purpose Districts**—Addresses that reside in five or more overlapping Special Purpose Districts. Addresses included in this count have a value of 5 or greater in the **NumberSPDsFound** output field.
- **Locations in User-Defined Districts**—Addresses that reside in at least one user-defined district. For more information on user-defined districts, see **User-Defined Boundary File** on page 384.
- **Locations in 1 User District**—Addresses that reside in one user-defined district. Addresses included in this count have a value of 1 in the **NumberUserBoundariesFound** output field.
- **Locations in 2 User Districts**—Addresses that reside in two overlapping user-defined districts. Addresses included in this count have a value of 2 in the **NumberUserBoundariesFound** output field.
- **Locations in 3+ User Districts**—Addresses that reside in three or more overlapping user-defined districts. Addresses included in this count have a value of 3 in the **NumberUserBoundariesFound** output field.

Generating a Summary Report

The following procedure describes to to add a report to a job.

1. In Enterprise Designer, on the bottom left side of the window under Palette, click **Reports**. A list of available reports appears.
2. Drag the Assign GeoTAX Info icon onto the canvas. You do not need to connect the icon to anything.
3. Double-click the report.
4. Select the stages that you want to contribute to the report.
5. If you want an output format other than PDF, such as html or txt, perform the following steps:
   a) Click the **Parameters** tab.

b) Clear the **Use default reporting options** check box and select the desired output format.

6. Click **OK**. On the toolbar, click the **Run Current Flow** (arrow) button. The `Execution Details` window displays.
7. When the Status field shows `succeeded`, under Reports, click **AssignGeoTAXInfo**. The Summary Report displays.

Match and Location Codes

Match Codes

The following table lists the Match Codes. The Match Codes indicate the portions of the address that matched or did not match to the reference file. If a match could not be made, the Match Code begins with E and the remaining digits indicate why the address did not match. The digits do not specifically refer to which address elements did not match, but rather why the address did not match. These fields are always included in the output from AssignGeoTAXInfo.

Table 151: Match Codes

columnName	Description
Ahh	Same as Shh, but indicates match to an alias name record or an alternate record.
Chh	Street address did not match, but located a street segment based on the input ZIP Code or city.
D00	Matched to a small town with P.O. Box or General Delivery only.
Ghh	Matched to an auxiliary file.
Hhh	House number was changed.
Qhh	Matched to USPS range records with unique ZIP Codes. CASS rules prohibit altering an input ZIP if it matches a unique ZIP Code value.
Rhh	Matched to a ranged address.
Shh	Matched to USPS data. This is considered the best address match, because it matched directly against the USPS list of addresses. S is returned for a small number of addresses when the matched address has a blank ZIP + 4.
Thh	Matched to a street segment record. Street segment records do not contain ZIP Code information. If you enter a ZIP Code, the application returns the ZIP Code you entered. If the input city and state has only one ZIP Code, the application returns that ZIP Code.
Uhh	Matched to USPS data but cannot resolve the ZIP + 4 code without the firm name or other information.
Xhhh	Matched to an intersection of two streets, for example, “Clay St & Michigan Ave.” The first hex digit refers to the last line information, the second hex digit refers to the first street in the intersection, and the third hex digit refers to the second street in the intersection. **Note:** The USPS does not allow intersections as a valid deliverable address

columnName	Description
Yhhh	Same as Xhhh, but an alias name record was used for one or both streets.
Z	No address given, but verified the provided ZIP Code.

Hex digit decoding

The following table contains the description of the hex digits for the Match Code values.

Code	In first hex position means:	In second and third hex position means:
0	No change in last line.	No change in address line.
1	ZIP Code changed.	Street type changed.
2	City changed.	Pre-directional changed.
3	City and ZIP Code changed.	Street type and pre-directional changed.
4	State changed.	Post-directional changed.
5	State and ZIP Code changed.	Street type and post-directional changed.
6	State and City changed.	Pre-directional and post-directional changed.
7	State, City, and ZIP Code changed.	Street type, pre-directional, and postdirectional changed.
8	ZIP + 4 changed.	Street name changed.
9	ZIP and ZIP + 4 changed.	Street name and street type changed.
A	City and ZIP + 4 changed.	Street name and pre-directional changed.
B	City, ZIP, and ZIP + 4 changed.	Street name, street type, and pre-directional changed.
C	State and ZIP + 4 changed.	Street name and post-directional changed.
D	State, ZIP, and ZIP + 4 changed.	Street name, street type, and post-directional changed.
E	State, City, and ZIP + 4 changed.	Street name, pre-directional, and postdirectional changed.
F	State, City, ZIP, and ZIP + 4 changed.	Street name, street type, pre-directional, and post-directional changed.

Match Error Codes

The following table describes the values returned when an error occurs or the application cannot find a match.

Table 152: Result Indicator Match Error Codes

columnName	Description
Ennn, where nnn =	Indicates an error, or no match. This can occur when the address entered does not exist in the database, or the address is badly formed and cannot be parsed correctly. The last three digits of an error code indicate

columnName	Description
	which parts of an address the application could not match to the database.
000	No match made.
001	Low level error.
002	Could not find data file.
003	Incorrect GSD file signature or version ID.
010	No city and state or ZIP Code found.
011	Input ZIP not in the directory.
012	Input city not in the directory.
013	Input city not unique in the directory.
014	Out of licensed area. Only occurs if using Group 1 licensing technology.
015	Record count is depleted and license has expired.
020	No matching streets found in directory.
021	No matching cross streets for an intersection match.
022	No matching segments.
023	Unresolved match.
024	No matching segments. (Same as 022.)
025	Too many possible cross streets for intersection matching.
026	No address found when attempting a multiline match.
027	Invalid directional attempted.
028	Record also matched EWS data, therefore the application denied the match.
029	No matching range, single street segment found.
030	No matching range, multiple street segments found.

Location Codes

The Location Codes indicate the methodology used to compute the geocode and may also provide some information about the quality of the geocode.

A Location Code of 'E' indicates a location code is not available. This usually occurs when you have requested ZIP Code centroids of a high quality, and one is not available for that match. It can occur infrequently when the Enterprise Tax Module does not have a 5-digit centroid location. An 'E' location code type may also be returned when the input address cannot be standardized and there is no input ZIP Code. In this case, do not assume the ZIP Code returned with the nonstandardized address is the correct ZIP Code because the Enterprise Tax Module did not standardize the address; therefore, the Enterprise Tax Module does not return geocoding or Census Block information.

Address Location Codes

Address location codes detail the known qualities about the geocode. An address location code has the following characters.

**Characters	Description	
1st character	Always an 'A' indicating an address location.	
2nd character	May be one of the following:	
	C	Interpolated address point location.
	G	GeoTAX Auxiliary fiel or Landmark Auxiliary file data location.
	I	Application infers the correct segment from the candidate records.
	P	Point-level data location.
	R	Location represents a ranged address.
	S	Location on a street range.
	X	Location on an intersection of two streets.
3rd and 4th characters	Digit indicating other qualities about the location.	

Table 153: Location Codes

**columnName	Description
ACn - Interpolated address point location	
AIn - The correct segment is inferred from the candidate records at match time.	
ASn - House range address geocode. This is the most accurate street interpolated geocode available.	
where n =	
0	Best location.
1	Street side is unknown. The Census FIPS Block ID is assigned from the left side; however, there is no assigned offset and the point is placed directly on the street.
2	Indicates one or both of the following: • The address is interpolated onto a TIGER segment that did not initially contain address ranges. • The original segment name changed to match the USPS spelling. This specifically refers to street type, predirectional, and postdirectional. **Note:** Only the second case is valid for non-TIGER data because segment range interpolation is only completed for TIGER data.
3	Both 1 and 2.
7	Placeholder. Used when starting and ending points of segments contain the same value and shape data is not available.
AGn - Indicates a geocode match to a GeoTAX Auxiliary or Landmark Auxiliary file,	

**columnName	Description
where n =	
0	The geocode represents the center of a parcel or building.
1	The geocode is an interpolated address along a segment.
2	The geocode is an interpolated address along a segment, and the side of the street cannot be determined from the data provided in the auxiliary file record.
3	The geocode is the midpoint of the street segment.
APnn - Indicates a point-level geocode match representing the center of a parcel or building,	
where nn =	
02	Parcel centroid Indicates the center of an assessor's parcel (tract or lot) polygon. When the center of an irregularly shaped parcel falls outside of its polygon, the centroid is manually repositioned to fall inside the polygon as closely as possible to the actual center.
04	Address point Represents field-collected GPS points with field-collected address data.
05	Structure centroid Indicates the center of a building footprint polygon, where the building receives mail or has telephone service. Usually a residential address consists of a single building. For houses with outbuildings (detached garages, shed, barns, etc.), only the residences have a structure point. Condominiums and duplexes have multiple points for each building. Larger buildings, such as apartment complexes, typically receive mail at one address for each building and therefore individual apartments are not represented as discrete structure points. Shopping malls, industrial complexes, and academic or medical center campuses where one building accepts mail for the entire complex are represented as one point. When addresses are assigned to multiple buildings within one complex, each addressed structure is represented by a point. If the center of a structure falls outside of its polygon, the center is manually repositioned to fall inside the polygon.
07	Manually placed Address points are manually placed to coincide with the midpoint of an assessor's parcel's street frontage at a distance from the center line.
08	Front door point Represents the designated primary entrance to a building. If a building has multiple entrances and there is no designated primary entrance or the primary entrance cannot readily be determined, the primary entrance is chosen based on proximity to the main access street and availability of parking.
09	Driveway offset point Represents a point located on the primary access road (most commonly a driveway) at a perpendicular distance of between 33- 98 feet (10-30 meters) from the main roadway.

**columnName	Description
10	Street access point Represents the primary point of access from the street network. This address point type is located where the driveway or other access road intersects the main roadway.
21	Base parcel point The Centrus point data includes individual parcels that may be "stacked". These stacked parcels are individually identified by their unit or suite number, and the Enterprise Tax Module is able to match to this unit number and return the correct APN. If an input address is for a building or complex, without a unit number, the "base" parcel information returns and will not standardize to a unit number or return additional information such as an APN.
3	The geocode is the midpoint of the street segment.
ARn - Ranged address geocode, where n =	
1	The geocode is placed along a single street segment, midway between the interpolated location of the first and second input house numbers in the range.
2	The geocode is placed along a single street segment, midway between the interpolated location of the first and second input house numbers in the range, and the side of the street is unknown. The Census FIPS Block ID is assigned from the left side; however, there is no assigned offset and the point is placed directly on the street.
4	The input range spans multiple USPS segments. The geocode is placed on the endpoint of the segment which corresponds to the first input house number, closest to the end nearest the second input house number.
7	Placeholder. Used when the starting and ending points of the matched segment contain the same value and shape data is not available.
AXn - Intersection geocode, where n =	
3	Standard single-point intersection computed from the center lines of street segments.
8	Interpolated (divided-road) intersection geocode. Attempts to return a centroid for the intersection.

Street centroid location codes

Street centroid location codes indicate the Census ID accuracy and the position of the geocode on the returned street segment. A street centroid location code has the following characters.

Character	Description
1st character	Always 'C' indicating a location derived from a street segment.
2nd character	Census ID accuracy based on the search area used to obtain matching street segment.
3rd character	Location of geocode on the returned street segment.

The following table contains the values and descriptions for the street centroid location codes.

Character position	Code	Description
2nd Character		
	B	Block Group accuracy (most accurate). Based on input ZIP Code.
	T	Census Tract accuracy. Based on input ZIP Code.
	C	Unclassified Census accuracy. Normally accurate to at least the County level. Based on input ZIP Code.
	F	Unknown Census accuracy. Based on Finance area.
	P	Unknown Census accuracy. Based on input City.
3rd Character		
	C	Segment centroid.
	L	Segment low-range endpoint.
	H	Segment high-range endpoint.

ZIP + 4 Location Codes

ZIP + 4® centroid location codes indicate the quality of two location attributes: Census ID accuracy and positional accuracy. A ZIP + 4 centroid location code has the following characters.

Character	Description
1st character	Always a 'Z' indicating a location derived from a ZIP centroid..
2nd character	Census ID accuracy.
3rd character	Location type.
4th character	How the location and Census ID was defined. Provided for completeness, but may not be useful for most applications.

The following table contains the values and descriptions for the ZIP + 4 location codes.

Character position	Code	Description
2nd Character		
	B	Block Group accuracy (most accurate).
	T	Census Tract accuracy.
	C	Unclassified Census accuracy. Normally accurate to at least the County level. Based on input ZIP Code.
3rd Character		
	5	Location of the Post Office that delivers mail to the address, a 5-digit ZIP Code centroid, or a location based upon locale (city). See the 4th character for a precise indication of locational accuracy.
	7	Location based upon a ZIP + 2 centroid. These locations can represent a multiple block area in urban locations, or a slightly larger area in rural settings.

Character position	Code	Description
	9	Location based upon a ZIP + 4 centroid. These are the most accurate centroids and normally place the location on the correct block face. For a small number of records, the location may be the middle of the entire street on which the ZIP + 4 falls. See the 4th character for a precise indication of locational accuracy.
4th Character		
	A	Address matched to a single segment. Location assigned in the middle of the matched street segment, offset to the proper side of the street.
	a	Address matched to a single segment, but the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	B	Address matched to multiple segments, all segments have the same Block Group. Location assigned to the middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	b	Same as methodology B except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	C	Address matched to multiple segments, with all segments having the same Census Tract. Returns the Block Group representing the most households in this ZIP + 4. Location assigned to t he middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	c	Same as methodology C except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	D	Address matched to multiple segments, with all segments having the same County. Returns the Block Group representing the most households in this ZIP + 4. Location assigned to the middle of the matched street segment with the most house number ranges within this ZIP + 4. Location offset to the proper side of the street.
	d	Same as methodology D except the correct side of the street is unknown. Location assigned in the middle of the matched street segment, offset to the left side of the street, as address ranges increase.
	E	Street name matched; no house ranges available. All matched segments have the same Block Group. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.

Character position	Code	Description
	F	Street name matched; no house ranges available. All matched segments have the same Census Tract. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.
	G	Street name matched (no house ranges available). All matched segments have the same County. Location placed on the segment closest to the center of the matched segments. In most cases, this is on the mid-point of the entire street.
	H	Same as methodology G, but some segments are not in the same County. Used for less than .05% of the centroids.
	I	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, and b. All centroids in this ZIP + 2 cluster have the same Block Group. Location assigned to the ZIP + 2 centroid.
	J	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, b, C, and c. All centroids in this ZIP + 2 cluster have the same Census Tract. Location assigned to the ZIP + 2 centroid.
	K	Created ZIP + 2 cluster centroid as defined by methodologies A, a, B, b, C, c, D, and d. Location assigned to the ZIP + 2 centroid.
	L	Created ZIP + 2 cluster centroid as defined by methodology E. All centroids in this ZIP + 2 cluster have the same Block Group. Location assigned to the ZIP + 2 centroid.
	M	Created ZIP+2 cluster centroid as defined by methodology E and F. All centroids in this ZIP + 2 cluster have the same Census Tract. Location assigned to the ZIP + 2 centroid.
	N	Created ZIP + 2 cluster centroid as defined by methodology E, F, G, and H. Location assigned to the ZIP + 2 centroid.
	O	ZIP Code is obsolete and not currently used by the USPS. Historic location assigned.
	V	Over 95% of addresses in this ZIP Code are in a single Census Tract. Location assigned to the ZIP Code centroid.
	W	Over 80% of addresses in this ZIP Code are in a single Census Tract. Reasonable Census Tract accuracy. Location assigned to the ZIP Code centroid.
	X	Less than 80% of addresses in this ZIP Code are in a single Census Tract. Census ID is uncertain. Location assigned to the ZIP Code centroid.
	Y	Rural or sparsely populated area. Census code is uncertain. Location based upon the USGS places file.

Character position	Code	Description
	Z	P.O. Box or General Delivery addresses. Census code is uncertain. Location based upon the Post Office location that delivers the mail to that address.

Geographic Centroid Location Codes

Geographic centroid location codes indicate the quality of two location attributes: the geographic location and area type.

Character	Description
1st character	Always 'G' indicating a location derived from a geographic centroid.
2nd character	Geographic area type.

The following table contains the values and descriptions for the geographic centroid location codes.

Character position	Code	Description
2nd Character		
	M	Municipality (city).
	C	County.
	S	State.

Payroll Tax Correspondence File

Using a Payroll Tax Correspondence File

Payroll system tax codes are proprietary codes used by some payroll tax applications to represent specific jurisdictions or combinations of jurisdictions. If you have licensed this option, you can use a payroll tax correspondence (PTC) file to determine the payroll system tax codes for a given address.

To use a payroll system tax code database, customize the PTC file and then install the file.

1. Customize the PTC file.

   After you receive your initial PTC file, modify it using a text editor of your choice. Specifically, you need to modify the following:

   - **Description**—A meaningful description of the code that represents business rules within your organization.
   - **Flags**—Flags indicate the payroll codes you want Assign GeoTAX Info to return. Possible flag values are:
   - **N**—No, do not return this payroll tax code. Continue searching the PTC file for other matching records. A blank flag has the same effect as "N".
   - **D**—Done. Return this payroll system tax code and stop searching the PTC file for other matching records.
   - **Any other value**—Any other value indicates to return the code. Typically, the letter Y (for "yes") is used. Use other flags if you like. For example, if there is an area with two codes, one for a work location and the other for the worker's residence, you could use flags of "W" and "R" so that the output field PTCn.PayrollFlag returned by AssignGeoTAX Info indicates the type of location.

   The following table shows how to use flags.

Note: Each record in the PTC file can contain up to six payroll system tax codes and their associated descriptions and flags. In the following example, the first three codes are on the first record and the second three are on the second. See **Table 155: Payroll System Tax Code File Layout** on page 436 for the layout of the PTC file.

Table 154: Example PTC File

Record	Description	Payrolll System Tax Code	Flag
1	HARBORCREEK TWP	123ABC	D
	HARBOR CREEK SD	456DEF	D
	HARBORCREEK TWP (M + SD)	789GHI	N
2	DAYTON BORO	592UID	Y
	ARMSTRONG SD	143XMA	Y
	DAYTON BORO (M + SD)	592JKT	N

In this example, the first record in the PTC file contains tax codes for a municipality called Harbor Creek Township and a school district called Harbor Creek School District. There is a separate code that represents points that are in both the Harbor Creek Township and the Harbor Creek School District. The second record in the PTC file contains codes for Dayton, a school district called Armstrong, and points located in both Dayton and the Armstrong School District.

This example returns the payroll system tax codes for the individual taxing jurisdictions for a given address, not the codes that represent the combined jurisdictions.

For addresses located in both the Harbor Creek Township and Harbor Creek School District, AssignGeoTAXInfo returns the following:

- PTC1.PayrollDescription=HARBORCREEK TWP
- PTC1.PayrollCode=123ABC
- PTC1.PayrollFlag=D
- PTC2.PayrollDescription=HARBOR CREEK SD
- PTC2.PayrollCode=456DEF
- PTC2.PayrollFlag=D

For addresses located in both Dayton and the Armstrong School District, AssignGeoTAXInfo returns the following:

- PTC1.PayrollDescription=DAYTON BORO
- PTC1.PayrollCode=592UID
- PTC1.PayrollFlag=Y
- PTC2.PayrollDescription=ARMSTRONG SD
- PTC2.PayrollCode=143XMA
- PTC2.PayrollFlag=Y
- For addresses located in just Dayton but not the Armstrong School District, AssignGeoTAXInfo returns the code 592UID and its associated flag and description. For addresses located in just the Armstrong School District but not Dayton, AssignGeoTAXInfo returns the code 143XMA and its associated flag and description.

Note: For a listing of output fields returned for payroll system tax codes, see **Payroll System Tax Code** on page 380.

The following table describes the PTC file layout.

Table 155: Payroll System Tax Code File Layout

Position	Length	Name
1	9	(required) Key Value. Lookup key into the file. Must be one of the following. • 5 character jurisdiction ID from the Pay.txb boundary file. This file contains polygons for special areas, such as school districts. • 9 character GNIS code. • 5 character county key made up of the State and County FIPS codes. • 2 character State FIPS code.
10	2	(optional) State
12	30	(optional) County
The payroll code and payroll description fields are paired. Each pair has a corresponding flag field. The flag field determines if GeoTAX returns data for the corresponding payroll code and description fields. GeoTAX does not return the fields if the flag is N or blank. This lets you create meaningful flags based on your business rules.		
42	15	Payroll code 1
57	40	Payroll description 1
97	15	Payroll code 2
112	40	Payroll description 2
152	15	Payroll code 3
167	40	Payroll description 3
207	15	Payroll code 4
222	40	Payroll description 4
262	15	Payroll code 5
277	40	Payroll description 5
317	15	Payroll code 6
332	40	Payroll description 6
372	1	Flag 1
373	1	Flag 2
374	1	Flag 3
375	1	Flag 4
376	1	Flag 5
377	1	Flag 6

2. Install the PTC file. For instructions on installing the PTC file, see the *Spectrum™ Technology Platform Installation Guide*

PTC file updates are provided at intervals defined in your contract. When you receive the PTC update files, enter flags for the modified records and then run Enterprise Tax Module database load utility to merge the updated PTC file with your existing file. For instructions, see "Installing Payroll Tax Correspondence Files" in the *Spectrum™ Technology Platform Installation Guide*. This process maintains the existing user-defined flags you have already associated with the PTC data.

User-Defined Data Files

Creating a Landmark Auxiliary File

The Landmark Auxiliary File provides you a way to specify customized address information in your input records. The recommended primary use of this file is to match to your company's non-address locations such as well heads, transmission towers or any other descriptive location. The latitude/latitude that is part of the input needed to build this file allows companies to automatically keep track of any jurisdictional changes that affect these unique locations. Matching to this file requires that the input record information match exactly to the Landmark file contents. This also applies to street records if you choose to enter them in the Landmark file.

Landmark Auxiliary file requirements

The Landmark Auxiliary file requirements include:

- The file must be a fixed-width text file: On Windows and UNIX, the text file must be ASCII
- The file must have a .gax extension on Windows and UNIX
- The file must have less than 500,000 records
- The file must follow the column field order and lengths specified in **Landmark File Layout**.

Record types

You can include two types of records in your Landmark Auxiliary file: Landmark and Street Records.

A **Landmark record** represent a single site. To be a valid landmark record, the record must have the following fields:

- ZIP Code
- Name of the landmark - placed in the street name field
- Beginning latitude of the landmark
- Beginning longitutde of the landmark

In addition, a Landmark record may NOT have the following fields:

- Street type abbreviation
- Pre-directional abbreviation
- Post-directional abbreviation
- Low house number
- High house number

A **Street record** contains a range of one or more addresses on a street. To be a valid street record the record must have the following fields:

- ZIP Code
- Street name
- Street type abbreviation, if part of the address
- Pre-directional abbreviation, if part of the address
- Post-directional abbreviation, if part of the address
- Low house number within the street segment
- High house number within the street segment
- Beginning longitude of the street segment
- Beginning latitude of the street segment

In addition, a Street record may NOT have the following fields:

- Secondary address information, such as unit numbers
- Mailstops

- Private mail boxes (PMBs)

During processing ignores any record that does not comply with the preceding requirements.

Landmark Auxiliary file organization

You must comply with the following organizational rules when creating your Landmark Auxiliary file.

- Use semicolons in the first column to indicate a row is a comment, not a data record; Enterprise Tax Module ignores rows that begin with a semicolon.
- Order the records within the file by descending ZIP Code then descending street name for optimal performance.
- All records must represent one or both sides of a street.
- All records must represent segments that are straight lines. Records cannot represent a non-straight segment.
- If house numbers are present in the record, the house number range must be valid according to USPS rules documented in Publication 28.
- The numeric fields, such as ZIP Codes, must contain all numbers.
- Latitude and longitude values must be in millionths of decimal degrees.
- Records cannot contain PO Box addresses.

Default values

Enterprise Tax Module uses the following defaults if you do not include the values in the Landmark Auxiliary file:

- House number parity = B (both odds and evens)
- Segment direction = F (forward) or A (ascending), these are interchangeable.
- Side of street = U (unknown)

Landmark Auxiliary file layout

Table 156: Landmark Auxiliary File Layout

Field	Description	Required For Street Segment Match	Required For Landmark Match	Requires Exact Match	Length	Position
ZIP Code	5-digit ZIP Code	X	X	X	5	1-5
Street name	Name of the street or landmark	X	X	X	30	6-35
Street type abbreviation	Street type. Also called street suffix. See the USPS Publication 28 for a complete list of supported street types.			X	4	36-39
Predirectional	USPS street name predirectional abbreviation. Supported values are N, E, S, W, NE, NW, SE, and SW.			X	2	40-41
Postdirectional	USPS street name postdirectional abbreviations. Supported values are N, E, S, W, NE, NW, SE, and SW.					
Reserved	Reserved				4	44-47

Field	Description	Required For Street Segment Match	Required For Landmark Match	Requires Exact Match	Length	Position
Low house number	Low house number of the address range.	X			11	48-58
High house number	High house number of the address range.	X			11	59-69
House number parity[3]	Parity of the house number in the range: • E - Even • O - Odd • B - Both				1	70
Segment direction	Direction the house numbers progress along the segment from the viewpoint of the segment's starting coordinate: • F - Forward (default) or A - Ascending • R - Reverse or D - Descending				1	71
Reserved	Reserved				1	72
FIPS state	US government FIPS state code.				2	73-74
FIPS county	US government FIPS county code.				3	75-77
Census tract	US Census tract number.				6	78-83
Census block group	US Census block group number.				1	84
Census block ID	US Census block ID number.				3	85-87
Reserved	Reserved				5	88-92
State abbreviation	USPS state abbreviation				2	93-95
County name	Name of the county.				25	95-119
MCD code	Minor Civil Division code.				5	120-124
MCD name	Minor Civil Division name.				40	125-164
CBSA code	Core Based Statistical Area code.				5	165-169
CBSA name	Core Based Statistical Area name.				49	170-218
Reserved	Reserved				5	219-223
City Name	City name. Overrides the city/state preferred city name upon a return.				40	224-263
Reserved	Reserved				237	264-500
User-defined data	User-defined data.				300	501-800

[3] For even and odd house number parity records, this specifies on which side of the street the house lays. For records containing both even and odd house numbers, the odd house numbers are on the specified side of the street, and the even house numbers are on the other side. This is a factor when using street offset.

Field	Description	Required For Street Segment Match	Required For Landmark Match	Requires Exact Match	Length	Position
Record ID Number	User-defined unique record identifier.				10	801-810
Side of street	Side of the street for the address: • L - Left side • R - Right side • B - Both sides • U - Unknown side (default) This is from the viewpoint of the segment's starting coordinate.				1	811
Beginning longitude	Beginning longitude of the street segment in millionths of degrees.	X	X		11	812-822
Beginning latitude	Beginning latitude of the street segment in millionths of degrees.	X	X		10	823-832
Ending longitude	Ending longitude of the street segment in millionths of degrees.				11	833-843
Ending latitude	Ending latitude of the street segment in millionths of degrees.				10	844-853

Matching to the Landmark Auxiliary file

Enterprise Tax Module performs the following steps when matching an input address to a Landmark Auxiliary file.

1. Enterprise Tax Module determines if there is a Landmark Auxiliary file present. If more than one Landmark Auxiliary file is present, Enterprise Tax Module attempts to match against the first file. Enterprise Tax Module ignores any additional Landmark Auxiliary files for matching, regardless if Enterprise Tax Module found a match to the first auxiliary file.

   If a record within the Landmark Auxiliary files is invalid, Enterprise Tax Module returns a message indicating the auxiliary file has an invalid record. Enterprise Tax Module continues to process input addresses against the Landmark Auxiliary file, but will not match to the invalid auxiliary file record.

2. If the Landmark Auxiliary file is present, Enterprise Tax Module first attempts to match to it. If more than one type of auxiliary file is present, the search order is:

   - Landmark Auxiliary file
   - User Auxiliary file
   - State-supplied file
   - GeoTAX Auxiliary file

   Note: Enterprise Tax Module only matches your input address to your Landmark Auxiliary file if there is an exact match. Therefore, your input address list should be as clean as possible; free of misspellings and incomplete addresses.

3. If Enterprise Tax Module finds an exact record match to the Landmark Auxiliary file, it standardizes the match to USPS regulations and returns the output of the auxiliary file match.

Note: You cannot update the Landmark Auxiliary file while Enterprise Tax Module is running. If you want to update the auxiliary file, you need to terminate Enterprise Tax Module before attempting to replace or edit the file.

Record type matching rules

When attempting a match against the Landmark Auxiliary file, GeoTAX abides by the following rules:

Landmark record match

- The input data must contain both a ZIP Code and address line, and they must exactly match the values on the auxiliary record.
- The input address cannot have any other data, such as a house number, unit number, or Private Mail Box (PMB).

Note: Enterprise Tax Module only matches the ZIP Code against the auxiliary file. GeoTAX does not verify that the ZIP Code of the input address record is correct for the city and state. You should validate this information in your input address before processing against the auxiliary file.

Street record match

- The input house number must fall within or be equal to the low and high house number values of the auxiliary record.
- The input house number must agree with the parity of the auxiliary record.
- The input ZIP Code must exactly match the ZIP Code of the auxiliary record.

Creating a User-Defined Auxiliary File

To enter data in a user-defined auxiliary file, you can use your own editor or data entry program. If you are running on Windows, you can use the Auxiliary File Editor available on the *GeoTAX Utilities CD*. The *GeoTAX Utilities CD* can be downloaded from the **www.g1.com/Support** web site. Select the **GeoTAX** product page; the utilities download and document links are provided on the **Subcomponents** panel.

Sort the records by ascending ZIP Code, street name, street type, directional, low house range, and descending high house range to minimize the time required to street-level match.

The auxiliary file has a block size of 8K, key length of 47 bytes, and record length of 800 bytes. The sample file `SEQAUX` illustrates the layout of the file.

The following table describes the layout of the auxiliary file. After you create a file with this layout, install the file using the Enterprise Tax Module database load utility. For instructions, see the *Spectrum™ Technology Platform Installation Guide*.

Table 157: User-Defined Auxiliary File Layout

Position	Field Name	Description	Length
Input Key Area (Must be Unique)			
1-5	G1GTAUX-ZIP-CODE	ZIP Code	5
6-35	G1GTAUX-STREETNAME	Street name	30
36-39	G1GTAUX-STREET-TYPE	Street type	4
40-41	G1GTAUX-PRE-DIR	Predirectional	2
42-43	G1GTAUX-POST-DIR	Postdirectional	2

Position	Field Name	Description	Length
44-47	G1GTAUX-SEQ	Sequence number, from 0001-9999	4
Output Area			
48-58	G1GTAUX-OUTPUTRANGE-FROM	First number in the house number range (right adjusted, blankfilled)	11
59-69	G1GTAUX-OUTPUTRANGE-TO	Last number in the house number range (right adjusted, blankfilled)	11
70	G1GTAUX-OUTPUTODD-EVEN	• O = Odd • E = Even	1
71	G1GTAUX-OUTPUTASC-DESC	• A = Ascending • D = Descending	1
72	Reserved		
73-74	G1GTAUX-STATE-CODE	FIPS State Code	2,0
75-77	G1GTAUX-COUNTYCODE	FIPS County Code	3,0
78-83	G1GTAUX-CENSUSTRACT	6-digit Census Tract number	6,0
84	G1GTAUX-BLOCKGROUP	1-digit Block Group	1
85-92	Reserved		8
93-94	G1GTAUX-STATE-ABBV	USPS state abbreviation	2
95-119	G1GTAUX-COUNTYNAME	County name	25
120-124	G1GTAUX-MCD-CODE	5-digit MCD Code	5,0
125-164	G1GTAUX-MCD-NAME	MCD Name	40
165-168	G1GTAUX-MSA-CODE	4-digit MSA Code	4,0
169-218	G1GTAUX-MSA-NAME	MSA Name	50
219-223	G1GTAUX-PLACE-CODE	5-digit Place Code	5,0
224-263	G1GTAUX-PLACENAME	Place Name	40
264-265	G1GTAUX-PLACECLASS-CODE	2-digit Place Class Code	2
266	G1GTAUX-PLACE-INCFLAG	• U - Unincorporated • I - Incorporated	1
267-273	G1GTAUX-PLACE-LASTANNEXED	Place last annexed date, in the format MM/YYYY	7
274-280	G1GTAUX-PLACE-LASTUPDATED	Place last updated date, in the format MM/YYYY	7

Position	Field Name	Description	Length
281-287	G1GTAUX-PLACE-LASTVERIFIED	Place last verified date, in the format MM/YYYY	7
288-296	G1GTAUX-PLACE-GNIS	GNIS Code	9
297-500	Reserved		204
501-800	G1GTAUX-AUX-AREA	User-defined data	300

Creating a User-Defined Boundary File

User-defined boundary files define areas of interest to your organization, such as sales territories or insurance rating territories. AssignGeoTAXInfo uses this data to determine if an address falls within an area of interest. For example, you can create a boundary file that defines your sales territories. AssignGeoTAXInfo can then determine the tax jurisdictions that apply to an address and the sales territory of that address.

Note: User-defined boundary files in AssignGeoTAXInfo allow you to perform basic "point-in-polygon" spatial analysis. Point In Polygon provides additional point-in-polygon features.

1. Create an ESRI shapefile (.SHP) or MapInfo data interchange format file (.MIF) with the boundaries you want. Your .SHP or .MIF file can have up to three columns of user-defined data. The first and second columns are 10 bytes long and the third column is 50 bytes long. For information on .SHP and .MIF files, see **User-Defined Centrus Databases**.
2. Use the Boundary File Conversion utility to convert your .SHP or .MIF file to a .TXB file. The Boundary File Conversion utility is available on the *GeoTAX Utilities CD*. The *GeoTAX Utilities CD* can be downloaded from the **www.g1.com/Support** web site. Select the **GeoTAX** product page; the utilities download and document links are provided on the **Subcomponents** panel.
3. Load the User-Defined Boundary file using the appropriate database load procedure based on your platform. The filename that should be entered for the User-Defined Boundary file is *usr.txb*.

GeoConfidence Module

GeoConfidence Module

The GeoConfidence Module is used to determine the probability that an address or street intersection is within a given area. The module takes an address or intersection's location (determined by Geocode US Address), converts that location to a point, line, or polygon (depending on the precision of the match), then compares that shape with a database of known shapes to see if the two overlap, and the percentage overlap. For example, you could use the GeoConfidence Module to make decisions on a flood zone rating based on how much overlap there is between an address's location and the flood zone data. Anything greater than a 95% overlap with a 100-year flood zone may indicate that the address is in the flood zone. Conversely, anything less than 95% could cause your business process to send the address to exception processing that might include a manual review.

An address or intersection can be geocoded to a point, an address along a street segment (an array of street segment points), ZIP + 4 centroid, ZIP + 2 centroid, or ZIP Code centroid (polygons). You can use these shapes (points, lines, or polygons) to compare with other shapes to determine overlap, which can be used to determine a risk or probability.

Different geoconfidence polygons are generated depending on the GeoConfidence result returned by the Enterprise Geocoding Module. Refer to the Enterprise Geocoding Module documentation for more information about the GeoConfidence information returned by the Enterprise Geocoding Module.

The GeoConfidence Module supports U.S. locations only.

Note: GeoConfidence uses services provided by the Enterprise Geocoding and Location Intelligence modules.

GeoConfidence Stages

GeoConfidence deploys three dataflows that you can modify in Enterprise Designer. Each dataflow consists of various components that were installed with the Enterprise Geocoding and Location Intelligence modules.

For information about each component in the installed dataflows, see the relevant component chapter in the *Spectrum™ Technology Platform User's Guide*.

The names of the dataflows are:

- **GeoConfidenceSurface** This is the dataflow that creates the geoconfidence surface that can be used for further analysis. The input is the GeoConfidence information that is returned from the Enterprise Geocoding Module. Currently, only the Geocode US Address stage can return this information.
- **CreatePointsConvexHull** This is a subflow that is used by the GeoConfidenceSurface template. You should not need to make any changes to this subflow.
- **FloodRiskAnalysis** This is an example dataflow.

GeoConfidence Databases

GeoConfidence uses the same databases as the Enterprise Geocoding and Location Intelligence modules.

For information about how to add these databases, see the *Spectrum™ Technology Platform Administration Guide*.

In addition to these databases, the GeoConfidence Module includes a database of ZIP Code polygons. This is used by GeoConfidenceSurface.

GeoConfidenceSurface

GeoConfidenceSurface returns geoconfidence polygons (also called surfaces) based on the quality of the geocode information generated by the Enterprise Geocoding Module. With the geoconfidence polygons generated, you can then overlap this polygon with other spatial data to determine a risk or probability.

This service is used by the GeoConfidence Module's FloodZoneAnalysis dataflow template.

Note: GeoConfidence uses services provided by the Enterprise Geocoding and Location Intelligence modules.

Input

The input fields for GeoConfidenceSurface are the output fields returned by the GeoConfidence output category of the Enterprise Geocoding Module. These fields are described below.

columnName Response Element	Max. Field Length (bytes)	Description
GeoConfidenceCode	13	The value returned in this field indicates which geoconfidence surface type has been returned.

columnName Response Element	Max. Field Length (bytes)	Description
		Possible values are: **INTERSECTION** A geocode point for the intersection of two streets. **ADDRESS** An array of street segment points representing the street segment where the address is located. **POINT** If the geocoder was able to match the address using point data, the point geometry where the address is located. **POSTAL1** A geocode point for the ZIP centroid. **POSTAL2** An array of points for all street segments in the ZIP + 2 in which the address is located. **POSTAL3** An array of points for street segments in the ZIP + 4 in which the address is located. **ERROR** An error has occurred.
StreetSegmentPoints	1024	An array of latitude/longitude values that represent the street segment points. **Note:** This field contains values only if the `GeoConfidenceCode` field returns a value of `ADDRESS`, `POSTAL2`, or `POSTAL3`.
GeoConfidenceCentroidLatitude	11	The latitude of the centroid of the geoconfidence polygon.
GeoConfidenceCentroidLongitude	12	The longitude of the centroid of the geoconfidence polygon.

Output

The GeoConfidenceSurface output field contains the geoconfidence polygon.

columnName	Description
Geometry	A geoconfidence polygon that represents the returned geometry.

Customizing the GeoConfidence Module

The GeoConfidence Module deploys three dataflow templates that you can modify in Enterprise Designer. Each dataflow consists of various components that were installed with the Enterprise Geocoding and Location Intelligence modules.

The names of the dataflow templates are:

- **GeoConfidenceSurface** This is the template that creates the geoconfidence surface that can be used for further analysis. The input is the GeoConfidence information that is returned from the Enterprise Geocoding Module. Currently, only the Geocode US Address stage can return this information. To customize this template, you must at minimum specify the 5-digit ZIP Code spatial source in the ZIP stage (Query Spatial).
- **CreatePointsConvexHull** This is a subflow that is used by the GeoConfidenceSurface template. You should not need to make any changes to this subflow.
- **FloodRiskAnalysis** This is an example template. To customize this template, you must at minimum specify Flood spatial source in the Find Nearest stage.

Universal Addressing Module

Universal Addressing Module

The Universal Addressing Module is an address quality module that can standardize and validate addresses, improving the deliverability of mail. The Universal Addressing Module can ensure that your address data adheres to quality standards established by the postal authority. An address that adheres to these standards is more likely to be delivered in a timely manner. In addition, mailers who follow these standards can qualify for significant postage discounts. For information on discounts for U.S. mail, refer to the USPS *Domestic Mail Manual* (*DMM*) available at **www.usps.com**. For information on discounts for Canadian mail, refer to the Canada Post website at **www.canadapost.ca**. For information on discounts for Australian mail, refer to the Australia Post website at **www.auspost.com.au**.

The Universal Addressing Module can be used in batch mode, realtime mode, or as a hosted service, depending on which option you have licensed. The batch version of the Universal Addressing Module is CASS Certified™ by the USPS®. It is also AMAS certified by Australia Post.

The Universal Addressing Module is one of two address quality modules available for Spectrum™ Technology Platform. The other address quality module, the Address Now Module, provides enhanced support for addresses outside the U.S. and Canada, including validation for more countries and double-byte support. If you have a large amount of international address data, you may want to consider using the Address Now Module for address standardization and validation.

Universal Addressing Components

The Universal Addressing Module consists of the following components. These components can work with U.S., Canadian, Australian, and international addresses as long as you are licensed for the appropriate database (if you are running Universal Addressing in your own environment) or hosted service (if you are utilizing Universal Addressing through the Pitney Bowes Software hosted services).

- **AutoCompleteLoqate**—Offers real-time entry of address data and returns instant results based on each character entered into the form, ensuring only accurate data is entered into the database.
- **GetCandidateAddresses**—Returns a list of possible matches for a given address.
- **GetCandidateAddressesLoqate**—Returns a list of possible matches for a given address using a Loqate engine and database.
- **GetCityStateProvince**—Returns the city and state/province for a given postal code.
- **GetCityStateProvinceLoqate**—Returns the city and state/province for a given postal code using a Loqate engine and database.
- **GetPostalCodes**—Returns the postal codes for a given city.
- **GetPostalCodesLoqate**—Returns the postal codes for a given city using a Loqate engine and database.
- **ValidateAddress**—Standardizes and validates addresses using U.S., Canadian, and international postal data.
- **ValidateAddressAUS**—Standardizes and validates addresses using Australian postal data.

- **ValidateAddressGlobal**—ValidateAddressGlobal provides enhanced address standardization and validation for addresses outside the U.S. and Canada. ValidateAddressGlobal can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you process a significant number of addresses outside the U.S. and Canada, you should consider using ValidateAddressGlobal.
- **ValidateAddressLoqate**—ValidateAddressLoqate standardizes and validates addresses using postal authority address data. ValidateAddress Loqate can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names, and so on.

Universal Addressing Databases

The Universal Addressing Module uses a number of required and optional databases. The databases are installed on the Spectrum™ Technology Platform server. Some of the databases are available by subscription from Pitney Bowes Software and are updated monthly or quarterly. Others are licensed from the USPS®. The following table lists the Universal Addressing databases.

Table 158: Universal Addressing Module Databases

Database Name & Description	Required or Optional	Supplier
U.S. Postal Database The U.S. Postal Database is in a Pitney Bowes proprietary format. It contains every house number range in the United States and is updated on a monthly basis. The database files contain the following information: • ZIP + 4® Code • Standardized address elements • City and state information The U.S. Postal Database also contains the data needed to perform Enhanced Street Matching (ESM) and All Street Matching (ASM). ESM and ASM apply extra matching logic to any input address that is not matched through the regular address validation process.	Required for U.S. address processing	Pitney Bowes Software monthly subscription
Canadian Postal Database The Canadian Postal database is in Pitney Bowes Software proprietary format. The database files contain the following information: • Postal code • Standardized address elements • Municipality and province information	Required for Canadian address processing	Pitney Bowes Software monthly subscription
Australia Post Postal Address File Database The Postal Address File is part of Australia Post's Address Matching Approval System (AMAS) program. The database file contains the following information: • Postal code • Standardized address elements	Required for Australian address processing	Pitney Bowes Software monthly subscription

Database Name & Description	Required or Optional	Supplier
International Postal Database The International Postal Database is a collection of postal address data from around the world. Data from each country is categorized according to the level of data available. The categories are: • **Category A**—Enables the validation and correction of an address's postal code, city name, state/county name, street address elements, and country name. • **Category B**—Enables the validation and correction of an address's postal code, city name, state/county name, and country name. It does not support the validation or correction of street address elements. • **Category C**—Enables the validation and correction of the country name, and the validation of the format of the postal code.	Required for International address processing	Pitney Bowes Software quarterly subscription
DPV® Database The Delivery Point Validation database allows you to check the validity of an individual mailing address in the U.S. The DPV database enhances the U.S. Postal database's ability to validate mailing addresses. **Note:** The DPV database also contains the data required for Commercial Mail Receiving Agency (CMRA) processing. Each time an edition of the U.S. Postal database is released, a corresponding edition of the DPV database is released. Although USPS licensing allows the use of the U.S. Postal database beyond the expiration date (with certain restrictions), DPV lookups may not be performed after the expiration date of the DPV database. USPS licensing prohibits using DPV data for the generation of addresses or address lists. To prevent the generation of address lists, the DPV database contains "false positive records." False positive records are artificially manufactured addresses. For each negative response that occurs in a DPV query, a query is made to the False/Positive table in the DPV database. A match to this table will stop DPV processing. USPS licensing also prohibits exporting the DPV data outside the United States.	Optional, but required for CASS Certified™ processing; U.S. addresses only	Pitney Bowes Software monthly subscription
eLOT® Database The Enhanced Line of Travel (eLOT) database is a U.S. address database that ensures that Enhanced Carrier Route mailings are sorted as close as possible to the actual delivery sequence. the eLOT database is required for certain types of postal discounts. You will receive monthly updates to your eLOT database on the same media as the U.S. Postal database.	Optional; U.S. addresses only	Pitney Bowes Software monthly subscription

Database Name & Description	Required or Optional	Supplier
You must install the U.S. Postal database and eLOT database from the same month (i.e., September eLOT data must be processed with a September U.S. Postal database). If the U.S. Postal database and the eLOT database are not from the same month, there may be ZIP + 4® Codes for which eLOT numbers cannot be assigned. The ZIP Code™, ZIP + 4 Code, carrier route code, and the delivery point of an address must be provided to assign a eLOT code.		
EWS Database The Early Warning System (EWS) database prevents address validation errors that can result due to a delay in postal data reaching the U.S. Postal database. The EWS database consists of partial address information limited to the ZIP Code™, street name, pre- and post-directionals, and a suffix. For an address record to be EWS-eligible, it must be an address not present on the most recent monthly production U.S. Postal database. The USPS® refreshes the EWS file on a weekly basis (Thursdays). You can download the EWS file from the USPS® website at **ribbs.usps.gov**.	Optional; U.S. addresses only	Download for free from USPS® website
LACS^{Link}® Database The LACS^{Link} database allows you to correct addresses that have changed as a result of a rural route address converting to street-style address, a PO Box renumbering, or a street-style address changing. USPS licensing prohibits using LACS^{Link} for the generation of addresses or address lists. To prevent the generation of address lists, the LACS^{Link} database contains "false positive records." False positive records are artificially manufactured addresses. For each negative response that occurs in a LACS^{Link} query, a query is made to the False/Positive table in the LACS^{Link} database. A match to this table will stop LACS^{Link} processing. USPS licensing also prohibits exporting the LACS^{Link} database outside the United States	Optional, but required for CASS Certified™ processing; U.S. addresses only	Pitney Bowes Software monthly subscription
RDI™ Database The Residential Delivery Indicator (RDI™) database contains data that can help you determine the best cost for shipping your packages. RDI is similar to DPV in that the RDI data is supplied as hash tables. However, RDI is a much simpler process than DPV in that the standard hash algorithm is only determined for the 9-digit and 11-digit ZIP Code™ rather than the entire address.	Optional; U.S. addresses only	License directly from USPS®
Suite$^{Link™}$ Database Suite$^{Link™}$ corrects secondary address information for U.S. business addresses whose secondary address information	Optional; U.S. addresses only	Pitney Bowes Software monthly subscription

Database Name & Description	Required or Optional	Supplier
could not be validated. If Suite[Link] processing is enabled, ValidateAddress attempts to match the value in the FirmName field to a database of known firm names. ValidateAddress then supplies the correct secondary address information.		

AutoCompleteLoqate

AutoCompleteLoqate offers real-time entry of address data for fast, accurate results. Users are returned instant results based on each character entered into the form, ensuring only accurate data is entered into the database.

Input

The following table lists the input for AutoCompleteLoqate.

Table 159: Input Format

columnName	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.
Country	The country code or name, in any of the following formats: • 2-digit ISO country code • 3-digit UPU Country code • English country name For a list of ISO codes, see **Country ISO Codes and Module Support**.
FirmName	The company or firm name.
PostalCode	The postal code for the address.
StateProvince	The state or province.

Options

Table 160: AutoCompleteLoqate Options

optionName	Description
Database.Loqate	Specifies the database to be used for address processing. Only databases that have been defined in the **Database Resources** panel in the Management Console are available.
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345
HomeCountry	Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses you process are in Canada, specify Canada. The valid country names are: Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent And The Grenadines, Samoa, San Marino, Sao Tome And Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia,

optionName	Description
	Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe
OutputCountryFormat	Specifies the format to use for the country name returned in the **Country** output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany". **E** Use English country names (default). **I** Use two-letter ISO abbreviation for the countries instead of country names. **U** Use Universal Postal Union abbreviation for the countries instead of country names.
OutputScript	Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native. **Input** Do not perform transliteration and provide output in the same script as the input (default). **Native** Output in the native script for the selected country wherever possible. **Latn** Use English values.
isPowersearchEnable	Reduces input time by up to 80% for 240 countries by using data in the form of an index file. When you conduct a search, the Loqate Engine will first look for the corresponding index. If present, the method will attempt to instantly return a list of candidate addresses. If the index is not present, or if the index does not return any results, the original search process will be triggered. **Note:** Powersearch can be performed when there are two and only two fields in the input file: the Country field and any one of the AddressLine fields. If you select this option and your input file contains additional fields, the original search process will automatically be triggered. To conduct its search, Auto Complete indexes use up to the first 10 characters for searches within the United States and up to the first 15 characters for searches within all other eligible countries. Spaces and punctuation are not factored into this count. Powersearch cannot be used for the following countries: Botswana, Ethiopia, India, Kazakhstan, Malaysia, Mongolia, Saint Kitts and Nevis, and San Marino. **Note:** You must have a valid license for Powersearch processing. If you select this option but are not licensed for Powersearch, or if your license has expired, you will receive an error.

optionName	Description
MaximumResults	The maximum number of addresses that AutoCompleteLoqate should return. The default is 10.
FailJobOnDataLicenseError	Specifies how you want Spectrum Technology Platform to respond when a data license error occurs. **Fail the job** Fail the entire job if a data license error occurs. **Fail the record** Fail the record(s) for which the data license error occurs and continue processing.

Output

The output from AutoCompleteLoqate is optional and corresponds directly to the fields you selected in the Output Fields section of the AutoCompleteLoqate Options dialog box.

Table 161: AutoCompleteLoqate Output

columnName	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.
Country	The three-character ISO 3116-1 Alpha-3 code for the country. For a list of ISO codes, see **Country ISO Codes and Module Support**.
FirmName	The firm name.
HouseNumber	The ending house number for the range in which the candidate address's house number falls.
PostalCode	The postal code.
PostalCode.AddOn	The last four digits of the ZIP + 4® Code.
ProcessedBy	Indicates which address coder processed the address. **LOQATE** The Loqate coder processed the address.
StateProvince	The state or province abbreviation.
Status	Reports the success or failure of the match attempt. **null** Success **F** Failure
Status.Code	The reason for failure, if there is one. • DisabledCoder • RequestFailed • NoLookupAddressFound

columnName	Description	
Status.Description	A description of the problem, if there is one.	
	Did not return multiples	The input address matched only one address in the database. AutoCompleteLoqate returns data only if multiple possible matches were found.
	Not able to look up the address pattern	AutoCompleteLoqate is not able to process the partial address.

AutoCompleteLoqate Sample Web Application

You can access a sample web application that demonstrates the Auto Complete Loqate functionality. When you enter a partial address, this application makes a call to the Auto Complete Loqate REST web service, which returns a suggested address.

Note: Prior to using this feature, you must add an Auto Complete Loqate database resource in Management Console and save the database resource in the Auto Complete Loqate Service.

1. Be sure the Spectrum™ Technology Platform server is running.
2. Open a web browser and go to: `http://<servername>:<port>/autocomplete`. For example, if your server is named "myserver" and it uses the default HTTP port 8080, you would go to: `http://myserver:8080/autocomplete`.

   Note: This site is best viewed in Internet Explorer 8.0 or later, Chrome, or Mozilla Firefox.

3. When the login screen appears, enter "**guest**" as the user name and leave the password field blank.
4. Press **OK**.
5. Select a country from the drop-down list.
6. Begin typing your address in any of the fields provided.
7. Select from the list of suggested addresses.
8. To begin a new call, click **Reset**, which will clear the fields you used in your previous call.

GetCandidateAddresses

GetCandidateAddresses returns a list of addresses that are considered matches for a given input address. GetCandidateAddresses returns candidate addresses only if the input address matches multiple addresses in the postal database. If the input address matches only one address in the postal database, then no address data is returned.

For addresses outside the U.S. and Canada, you may notice inconsistent results between the multiple matches returned by ValidateAddress and the results for that same address returned by GetCandidateAddresses. If you experience inconsistent results, it is likely because you set the performance tuning setting in ValidateAddress to a value other than 100. To obtain consistent results between GetCandidateAddresses and ValidateAddress, set the performance tuning option to 100.

Note: By default, GetCandidateAddresses does not match to individual house numbers. Rather, it uses house number ranges for each street. After GetCandidateAddresses has determined the street name, city name, state/province name, and postal code, it checks to make sure the input house number falls within one of the ranges of house numbers given for the matched street name. The same type of logic applies to unit numbers. If you want to determine that an individual house number is valid, you should use the ValidateAddress Delivery Point Validation (DPV) processing option. DPV processing is only available for U.S. addresses.

The Canadian coder contains a reverse lookup routine that takes as input a specific postal code and returns the street information stored in the database for that postal code. To use this function enter nothing but a Canadian postal code in the PostalCode field. See the second example to view the return from a sample postal code.

GetCandidateAddresses is part of the Universal Addressing Module.

Input

The following table lists the input for GetCandidateAddresses.

Table 162: Input Format

columnName	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line. Does not apply to U.S. and Canadian addresses.
AddressLine4	The fourth address line. Does not apply to U.S. and Canadian addresses.
AddressLine5	The fifth address line. Applies only to U.K. addresses. May contain street name, unit number, building number, and so on.
City	The city name.
StateProvince	The state or province. For U.S. addresses only, you may put the state in the City field instead of the StateProvince field.
PostalCode	The postal code for the address. For U.S. addresses this is the ZIP Code™ in one of the following formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999 **Note:** For Canadian addresses you can complete just this field and have candidate address data returned. For other countries, AddressLine1 and AddressLine2 must also be completed.
Country	The country code or name, in any of the following formats: • 2-digit ISO country code • 3-digit UPU Country code • English country name • French country name • German country name • Spanish country name For a list of ISO codes, see **Country ISO Codes and Module Support**.

columnName	Description
FirmName	The company or firm name.
USUrbanName	U.S. address urbanization name. Used primarily for Puerto Rico addresses.

Options

Table 163: GetCandidateAddresses Options

optionName	Description
PerformUSProcessing	Specifies whether or not to process U.S. addresses. If you enable U.S. address processing GetCandidateAddresses will attempt to retrieve candidate addresses for U.S. addresses. If you disable U.S. address processing, U.S. addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for U.S. address processing you must disable U.S. address processing in order for your jobs to complete successfully, regardless of whether or not they contain U.S. addresses. **Note:** You must have a valid license for U.S. address processing to successfully process U.S. addresses. If you enable U.S. address processing but are not licensed for this feature, or your license has expired, you will receive an error. **Y** Yes, process U.S. addresses (default). **N** No, do not process U.S. addresses.
Database.US	Specifies the database to be used for U.S. address processing. Only databases that have been defined in the **US Database Resources** panel in the Management Console are available.
PerformCanadianProcessing	Specifies whether or not to process Canadian addresses. If you enable Canadian address processing GetCandidateAddresses will attempt to retrieve candidate addresses for Canadian addresses. If you disable Canadian address processing, Canadian addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for Canadian address processing you must disable Canadian address processing in order for your jobs to complete successfully, regardless of whether or not they contain Canadian addresses. **Note:** You must have a valid license for Canadian address processing to successfully process Canadian addresses. If you enable Canadian address processing but are not licensed for this feature, or your license has expired, you will receive an error.

optionName	Description
	Y Yes, process Canadian addresses (default). **N** No, do not process Canadian addresses.
Database.Canada	Specifies the database to be used for Canadian address processing. Only databases that have been defined in the **Canadian Database Resources** panel in the Management Console are available.
PerformInternationalProcessing	Specifies whether or not to process international addresses (addresses outside the U.S. and Canada). If you enable international address processing GetCandidateAddresses will attempt to retrieve candidate addresses for international addresses. If you disable international address processing, international addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for international address processing you must disable international address processing in order for your jobs to complete successfully, regardless of whether or not they contain international addresses. **Note:** You must have a valid license for international address processing to successfully process international addresses. If you enable international address processing but are not licensed for this feature, or your license has expired, you will receive an error. **Y** Yes, process international addresses (default). **N** No, do not process international addresses.
Database.International	Specifies the database to be used for international address processing. Only databases that have been defined in the **International Database Resources** panel in the Management Console are available.
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output is in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output is in upper case. For example: 123 MAIN ST MYTOWN FL 12345
MaximumResults	The maximum number of candidate addresses that GetCandidateAddresses should return. The default is 10. The maximum is 10.
OutputShortCityName	For U.S. addresses, specifies whether or not to return the USPS®-approved abbreviation for the city, if there is one. The USPS® provides abbreviations for city names that are 14 characters long or longer. City abbreviations are 13 characters

optionName	Description
	or less and can be used when there is limited space on the mailing label. If there is no short city name for the city, then the full city name is returned. **Y** Yes, return the short city name. **N** No, do not return the short city name.
DualAddressLogic	(U.S. addresses only). Controls whether GetCandidateAddresses should return a street match or a PO Box/Rural Route/Highway Contract match when the address contains both street and PO Box/Rural Route/Highway Contract information. For more information, see **About Dual Address Logic** on page 484. **N** (Default) USPS® CASS™ regulations determine the address returned based on the following order of priority: **1.** PO Box **2.** Firm **3.** Highrise **4.** Street **5.** Rural Route **6.** General Delivery **S** Return a street match, regardless of the address line. **P** Return a PO Box match, regardless of the address line.
StreetMatchingStrictness	The strictness of the street name match (U.S. addresses only). **E** The input street name must match the database exactly. **T** The matching algorithm is "tight." **M** The matching algorithm is "medium" (default). **L** The matching algorithm is "loose".
FirmMatchingStrictness	The strictness of the firm name match (U.S. addresses only). **E** The input firm name must match the database exactly. **T** The matching algorithm is "tight." **M** The matching algorithm is "medium" (default). **L** The matching algorithm is "loose."
DirectionalMatchingStrictness	The strictness of the directional match. **E** The input directional must match the database exactly. **T** The matching algorithm is "tight." **M** The matching algorithm is "medium" (default).

optionName	Description
	L The matching algorithm is "loose."
PerformESM	Specifies whether or not to perform Enhanced Street Matching (ESM). ESM applies extra matching logic with additional data to any input address that is not matched through the regular address validation process. ESM applies to U.S. addresses only. **Y** Yes, perform ESM processing. **N** No, do not perform ESM processing (default).
AddressLineSearchOnFail	Specifies whether ValidateAddress will search address lines for the city, state/province, and postal code. This option enables ValidateAddress to search the AddressLine input fields for the city, state/province, postal code, and country when the address cannot be matched using the values in the City, StateProvince, and PostalCode input fields. Consider enabling this option if your input addresses have the city, state/province, and postal code information in the AddressLine fields. Consider disabling this option if your input addresses use the City, State/Province and PostalCode fields. If you enable this option and these fields are used, there is an increased possibility that ValidateAddress will fail to correct values in these fields (for example a misspelled city name). **Y** Yes, search the address line fields (default). **N** No, do not search the AddressLine fields.

Output

GetCandidateAddresses returns the following output.

Table 164: GetCandidateAddresses Output

columnName	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
AddressLine5	For U.K. addresses only. If the address was validated, the fifth line of the validated and standardized address. If the address could not be validated, the fifth line of the input address without any changes.
City	The city name.

columnName	Description
Country	The three-character ISO 3116-1 Alpha-3 code for the country. For a list of ISO codes, see **Country ISO Codes and Module Support**.
FirmName	The firm name.
HouseNumberHigh	The ending house number for the range in which the candidate address's house number falls.
HouseNumberLow	The beginning house number for the range in which the candidate address's house number falls.
HouseNumberParity	Indicates the numbering scheme for the house numbers between HouseNumberLow and HouseNumberHigh, as follows: **E** Only even values **O** Only odd values **B** Both
MatchLevel	For addresses outside the U.S. and Canada, identifies the match level for the candidate address. U.S. and Canadian addresses are always "A." One of the following: **A** The candidate matches the input address at the street level. **B** The candidate matches the input address at the state/province level.
PostalCode	The postal code. In the U.S. this is the ZIP Code™.
PostalCode.AddOn	The last four digits of the ZIP + 4® Code. U.S. addresses only.
RecordType	The type of address record, as defined by U.S. and Canadian postal authorities (U.S. and Canadian addresses only): • FirmRecord • GeneralDelivery • HighRise • PostOfficeBox • RRHighwayContract • Normal
RecordType.Default	Code indicating the "default" match: **Y** The address matches a default record. **null** The address does not match a default record.
StateProvince	The state or province abbreviation.
Status	Reports the success or failure of the match attempt. **null** Success **F** Failure
Status.Code	The reason for failure, if there is one. There is only one possible value: • DisabledCoder • RequestFailed

columnName	Description
Status.Description	A description of the problem, if there is one. **Did not return multiples** — The input address matched only one address in the database. GetCandidateAddresses only returns data if multiple possible matches were found. **Number of candidates is not greater than 1** — The input address matched more than one address in the database but no addresses were returned. **PerformUSProcessing disabled** — This value will appear if Status.Code=DisabledCoder. **PerformCanadianProcessing disabled** — This value will appear if Status.Code=DisabledCoder. **PerformInternationalProcessing disabled** — This value will appear if Status.Code=DisabledCoder.
UnitNumberHigh	The ending unit number for the range in which the candidate address's unit number falls.
UnitNumberLow	The beginning unit number for the range in which the candidate address's unit number falls.
UnitNumberParity	Indicates the numbering scheme for the unit numbers between UnitNumberLow and UnitNumberHigh, as follows: **E** — Only even values **O** — Only odd values **B** — Both
USUrbanName	The validated city urbanization name. Urbanization names are used primarily for Puerto Rico addresses.

GetCandidateAddressesLoqate

GetCandidateAddressesLoqate returns a list of addresses that are considered matches for a given input address. GetCandidateAddressesLoqate returns candidate addresses only if the input address matches multiple addresses in the postal database. If the input address matches only one address in the postal database, then no address data is returned. The Country input field is required; if this field is blank, no output will be returned.

Note: By default, GetCandidateAddressesLoqate does not match to individual house numbers. Rather, it uses house number ranges for each street. After GetCandidateAddressesLoqate has determined the street name, city name, state/province name, and postal code, it checks to make sure the input house number falls within one of the ranges of house numbers given for the matched street name. The same type of logic applies to unit numbers.

GetCandidateAddressesLoqate is part of the Universal Addressing Module.

Input

The following table lists the input for GetCandidateAddressesLoqate.

Table 165: Input Format

columnName	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.
Country	The country code or name, in any of the following formats: • 2-digit ISO country code • 3-digit UPU Country code • English country name For a list of ISO codes, see **Country ISO Codes and Module Support**. **Note:** This field is required. If this field is blank, no output will be returned.
FirmName	The company or firm name.
PostalCode	The postal code for the address. For U.S. addresses this is the ZIP Code™ in one of the following formats:
StateProvince	The state or province. For U.S. addresses only, you may put the state in the City field instead of the StateProvince field.

Options

Table 166: GetCandidateAddressesLoqate Options

optionName	Description
Database.Loqate	Specifies the database to be used for address processing. Only databases that have been defined in the Management Console are available.
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output is in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output is in upper case. For example: 123 MAIN ST MYTOWN FL 12345
HomeCountry	Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses

optionName	Description
	you process are in Canada, specify Canada. GetCandidateAddressLoqate uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields. The valid country names are: Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent And The Grenadines, Samoa, San Marino, Sao Tome And Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe
OutputCountryFormat	Specifies the format to use for the country name returned in the **Country** output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany". **E** Use English country names (default). **I** Use two-letter ISO abbreviation for the countries instead of country names. **U** Use Universal Postal Union abbreviation for the countries instead of country names.

optionName	Description
OutputScript	Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native. **Input** Do not perform transliteration and provide output in the same script as the input (default). **Native** Output in the native script for the selected country wherever possible. **Latn** Use English values.
MaximumResults	The maximum number of candidate addresses that GetCandidateAddressesLoqate should return. The default is 10. The maximum is 99.

Output

GetCandidateAddressesLoqate returns the following output.

Table 167: GetCandidateAddressesLoqate Output

columnName	Description
AddressLine1	The first address line.
AddressLine2	The second address line.
AddressLine3	The third address line.
AddressLine4	The fourth address line.
City	The city name.
Country	The three-character ISO 3116-1 Alpha-3 code for the country. For a list of ISO codes, see **Country ISO Codes and Module Support**.
FirmName	The firm name.
PostalCode	The postal code. In the U.S. this is the ZIP Code™.
PostalCode.AddOn	The last four digits of the ZIP + 4® Code. U.S. addresses only.
ProcessedBy	Indicates which address coder processed the address. **LOQATE** The Loqate coder processed the address.
StateProvince	The state or province abbreviation.
Status	Reports the success or failure of the match attempt. **null** Success **F** Failure
Status.Code	The reason for failure, if there is one. There is only one possible value: • RequestFailed

columnName	Description
Status.Description	A description of the problem, if there is one. There is only one possible value: **Did not return multiples** — The input address matched only one address in the database. GetCandidateAddressesLoqate only returns data if multiple possible matches were found.

GetCityStateProvince

GetCityStateProvince returns a city and state/province for a given input postal code.

Note: GetCityStateProvince works with U.S. and Canadian addresses only.

GetCityStateProvince is part of the Universal Addressing Module.

Input

The following table shows the input fields.

Table 168: GetCityStateProvince Input

columnName	Description
PostalCode	A U.S. ZIP Code™ or Canadian postal code in one of the following formats: 99999 99999-9999 A9A9A9 A9A 9A9

Options

Table 169: GetCityStateProvince Options

optionName	Description
PerformUSProcessing	Specifies whether or not to process U.S. addresses. If you enable U.S. address processing GetCityStateProvince will attempt to return the state for U.S. addresses. If you disable U.S. address processing, U.S. addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for U.S. address processing you must disable U.S. address processing in order for your jobs to complete successfully, regardless of whether or not they contain U.S. addresses. **Note:** You must have a valid license for U.S. address processing to successfully process U.S. addresses. If you enable U.S. address processing but are not licensed for this feature, or your license

optionName	Description
	has expired, you will receive an error. If you enable U.S. address processing but are not licensed for this feature, or your license has expired, you will receive an error. **Y** Yes, process U.S. addresses (default). **N** No, do not process U.S. addresses.
Database.US	Specifies the database to be used for U.S. address processing. Only databases that have been defined in the **US Database Resources** panel in the Management Console are available.
PerformCanadianProcessing	Specifies whether or not to process Canadian addresses. If you enable Canadian address processing GetCityStateProvince will attempt to return the province for Canadian addresses. If you disable Canadian address processing, Canadian addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for Canadian address processing you must disable Canadian address processing in order for your jobs to complete successfully, regardless of whether or not they contain Canadian addresses. **Note:** You must have a valid license for Canadian address processing to successfully process Canadian addresses. If you enable Canadian address processing but are not licensed for this feature, or your license has expired, you will receive an error. If you enable Canadian address processing but are not licensed for this feature, or your license has expired, you will receive an error. **Y** Yes, process Canadian addresses (default). **N** No, do not process Canadian addresses.
Database.Canada	Specifies the database to be used for Canadian address processing. Only databases that have been defined in the **Canadian Database Resources** panel in the Management Console are available.
OutputVanityCity	Specifies whether or not to include non-mailing city names in the output. A non-mailing city name is an alternate name for the primary city name. For example, Hollywood is a non-mailing city name for Los Angeles. **Y** Yes, include non-mailing city names. **N** No, do not include non-mailing city names (default).
MaximumResults	Specifies the maximum number of city-state/province pairs to return. The default value is 10.

Output

GetCityStateProvince returns the matching city and state/province for the input postal code as well as a code to indicate the success or failure of the match attempt. If more than one city/state or city/province matches the input postal code, multiple output records are returned.

Table 170: GetCityStateProvince Output

columnName	Description
City	The matched city name.
City.Type	The USPS® standardized city name type (U.S. addresses only). **V** Vanity (non-mailing) city name. **P** Primary. The city name is the primary mailing city name. **S** Secondary. The city name is an alternate city name but is acceptable. A city can have multiple secondary city names.
PostalCode	The input postal code.
ProcessedBy	Indicates which address coder processed the address. One of the following: **USA** The U.S. address coder processed the address. **CAN** The Canadian address coder processed the address.
StateProvince	The state or province abbreviation.
Status	Reports the success or failure of the match attempt. **null** Success **F** Failure
Status.Code	The reason for failure, if there is one. The only valid value is: • DisabledCoder • UnrecognizedPostalCode
Status.Description	The description of the failure. The valid values are: **Postal code not found** This value will appear if Status.Code=UnrecognizedPostalCode. **PerformUSProcessing disabled** This value will appear if Status.Code=DisabledCoder. **PerformCanadianProcessing disabled** This value will appear if Status.Code=DisabledCoder.

GetCityStateProvinceLoqate

GetCityStateProvinceLoqate returns a city and state/province for a given input postal code.

This stage is part of the Universal Addressing Module.

Input

The following table shows the input fields.

Table 171: GetCityStateProvinceLoqate Input

columnName	Description
Country	The country code or name, in any of the following formats: • 2-digit ISO country code • 3-digit UPU Country code • English country name For a list of ISO codes, see **Country ISO Codes and Module Support**.
PostalCode	The postal code for the address.

Options

Table 172: GetCityStateProvinceLoqate Options

columnName	Description / Valid Values
Database.Loqate	Specifies the database to be used for address processing. Only databases that have been defined in the **Database Resources** panel in the Management Console are available.
MaximumResults	The maximum number of addresses that GetCityStateProvinceLoqate should return. The default is 10.
OutputScript	Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native. **Input** — Do not perform transliteration and provide output in the same script as the input (default). **Native** — Output in the native script for the selected country wherever possible. **Latn** — Use English values.
FailJobOnDataLicenseError	Specifies how you want Spectrum Technology Platform to respond when a data license error occurs. **Fail the job** — Fail the entire job if a data license error occurs. **Fail the record** — Fail the record(s) for which the data license error occurs and continue processing.

Output

GetCityStateProvinceLoqate returns the matching city and state/province for the input postal code as well as a code to indicate the success or failure of the match attempt. If more than one city/state or city/province matches the input postal code, multiple output records are returned.

Table 173: GetCityStateProvinceLoqate Output

columnName	Description
City	The matched city name.
Country	The country in the format determined by what you selected in OutputCountryFormat: • ISO Code • UPU Code • English
PostalCode	The input postal code.
ProcessedBy	Indicates which address coder processed the address. **LOQATE** The Loqate coder processed the address.
StateProvince	The state or province abbreviation.
Status	Reports the success or failure of the match attempt. **null** Success **F** Failure
Status.Code	The reason for failure, if there is one. The only valid value is: • UnrecognizedPostalCode
Status.Description	The description of the failure. The only valid value is: **Postal code not found** This value will appear if Status.Code=UnrecognizedPostalCode.

GetPostalCodes

GetPostalCodes allows you to look up the postal codes for a particular city. The service takes a city, state, and country as input and returns the postal codes for that city. The input must be exactly correct in order to return postal codes.

Note: GetPostalCodes only works with U.S. addresses.

GetPostalCodes is part of the Universal Addressing Module.

Input

GetPostalCodes takes a city, state/province, and country as input.

Table 174: GetPostalCodes Input

columnName	Description
City	The city whose postal codes you want to look up. You may put the city and state in the City field. If you do this, you must leave the StateProvince field blank.

columnName	Description
	The total length of the City and StateProvince fields cannot exceed 100 characters.
StateProvince	The state or province of the city whose postal codes you want to look up. You may also put the state in the City field instead of the StateProvince field. The total length of the City and StateProvince fields cannot exceed 100 characters.
Country	The country code or name of the city whose postal codes you want to look up. The only valid value is US.

Options

Table 175: GetPostalCodes Options

optionName	Description
Database.US	Specifies the database to be used for postal code look-ups. Only databases that have been defined in the US Database Resources panel in the Management Console are available.
IncludeVanityCity	Specifies whether or not to include postal codes for the city's non-mailing city names. A non-mailing city name is an alternate name for the primary city name. For example, Hollywood is a non-mailing city name for Los Angeles. **Y** Yes, include postal codes for non-mailing city names. **N** No, do not include postal codes for non-mailing city names (default).
OutputCityType	Specifies whether or not to return the city type in the output. If enabled, the city type is returned in the City.Type field. **Y** Yes, include the city type in the output. **N** No, do not include the city type in the output (default).

Output

GetPostalCodes returns the postal codes for a specified city. Each postal code is returned in a separate record along with the data listed in the following table.

Table 176: GetPostalCodes Output

columnName	Description
City.Type	The USPS® city type (U.S. addresses only). The city type is determined by looking at the ZIP Code and the city name. For example, the city

columnName	Description
	Lanham MD has the postal codes 20703, 20706, and 20784. Lanham is the primary city in 20703 and 20706 but is a vanity city in 20784. This field column is only populated if OutputCityType=Y. The possible values are: **V** Vanity (non-mailing) city name. **P** Primary. The city name is the primary mailing city name. **S** Secondary. The city name is an alternate city name but is acceptable. A city can have multiple secondary city names.
PostalCode	A postal code in the specified city.
ProcessedBy	Because this service only works for U.S. addresses, ProcessedBy will always contain one value: USA.
Status	Reports the success or failure of the match attempt. **null** Success **F** Failure
Status.Code	Reason for failure, if there is one. One of the following: • CountryNotSupported • UnableToLookup
Status.Description	Description of failure. • Input country is not supported • Input city was blank • Input city & state / province was blank, or no match found • City-state mismatch (different spelling found, or city-state was a vanity name and vanity matching was not allowed, or city-state did not match ZIP Code)

GetPostalCodes Loqate

GetPostalCodesLoqate allows you to look up the postal codes for a particular city. The service takes a city, state, and country as input and returns the postal codes for that city. The input must be exactly correct in order to return postal codes.

GetPostalCodesLoqate is part of the Universal Addressing Module.

Input

GetPostalCodesLoqate takes a city, state/province, and country as input.

Table 177: GetPostalCodesLoqate Input

columnName	Description / Valid Values
City	The city whose postal codes you want to look up.

columnName	Description / Valid Values
	You may put the city and state in the City column. If you do this, you must leave the StateProvince column blank.
Country	The country code or name, in any of the following formats: • 2-digit ISO country code • 3-digit UPU Country code • English country name For a list of ISO codes, see **Country ISO Codes and Module Support**.
StateProvince	The state or province of the city whose postal codes you want to look up. You may also put the state in the City column instead of the StateProvince column.

Options

Table 178: GetPostalCodesLoqate Options

optionName	Description/Valid Values
Database.Loqate	Specifies the database to be used for postal code look-ups. Only databases that have been defined in the Management Console are available.
FailJobOnDataLicenseError	Specifies how you want Spectrum Technology Platform to respond when a data license error occurs. **Fail the job** — Fail the entire job if a data license error occurs. **Fail the record** — Fail the record(s) for which the data license error occurs and continue processing.

Output

GetPostalCodesLoqate returns the postal codes for a specified city. Each postal code is returned in a separate record along with the data listed in the following table.

Table 179: GetPostalCodesLoqate Output

columnName	Description / Valid Values
PostalCode	A postal code in the specified city.
ProcessedBy	Indicates which address coder processed the address. **LOQATE** — The Loqate coder processed the address.
Status	Reports the success or failure of the match attempt. **null** — Success **F** — Failure

columnName	Description / Valid Values
Status.Code	Reason for failure, if there is one. One of the following: • InvalidCountry • UnableToLookup
Status.Description	Description of failure. • Input country is not supported • Input city was blank • Input city & state / province was blank, or no match found

ValidateAddress

ValidateAddress standardizes and validates addresses using postal authority address data. ValidateAddress can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names, and more.

ValidateAddress also returns result indicators about validation attempts, such as whether or not ValidateAddress validated the address, the level of confidence in the returned address, the reason for failure if the address could not be validated, and more.

During address matching and standardization, ValidateAddress separates address lines into components and compares them to the contents of the Universal Addressing Module databases. If a match is found, the input address is *standardized* to the database information. If no database match is found, ValidateAddress optionally *formats* the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.

ValidateAddress is part of the Universal Addressing Module.

Input

ValidateAddress takes an address as input. All addresses use this format regardless of the address's country. See **Address Line Processing for U.S. Addresses** on page 475 for important information about how address line data is processed for U.S. addresses.

Table 180: Input Format

columnName	Format	Description
AddressLine1	String [50]	The first address line.
AddressLine2	String [50]	The second address line.
AddressLine3	String [50]	The third address line. Does not apply to Canadian addresses.
AddressLine4	String [50]	The fourth address line. Does not apply to Canadian addresses.
AddressLine5	String [50]	The fifth address line. Applies only to U.K. addresses. May contain street name, unit number, building number, and so on.

columnName	Format	Description
City	String [50]	The city name. For U.S. addresses only, you may put the city, state, and ZIP Code™ in the City field. If you do this, you must leave the StateProvince and PostalCode fields blank.
StateProvince	String [50]	The state or province. For U.S. addresses only, you may put the state in the City field instead of the StateProvince field.
PostalCode	String [10]	The postal code for the address in one of the following formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999 For U.S. addresses only, you may put the ZIP Code™ in the City field. For U.S. addresses only, if the city/state/ZIP Code™ is in the PostalCode field, ValidateAddress may parse the data and successfully process the address. For best results, put this data in the appropriate fields (City, StateProvince, and PostalCode).
Country	String [50]	The country code or name, in any of the following formats: • Two-character ISO 3116-1 Alpha-2 country code • Three-character ISO 3116-1 Alpha-3 country code • English country name • French country name • German country name • Spanish country name For a list of ISO codes, see **Country ISO Codes and Module Support**.
FirmName	String [50]	The company or firm name.
USUrbanName	String [50]	The U.S. address urbanization name. This is used primarily for Puerto Rico addresses.
CustomerID	String [9]	If this mailpiece uses a generic barcode, specify your USPS®-assigned customer ID in this field. The ValidateAddress generic barcode is used for mailpieces that use the OneCode ACS® service.
CanLanguage	String	For Canadian addresses only, indicates whether the address is in English or French, if the option CanFrenchFormat=T is used. If this field is blank, the address is formatted in English. If the field contains any non-blank value, the address is formatted

columnName	Format	Description
		in French. Note that addresses in Quebec are always formatted in French regardless of the value in this field.

Address Line Processing for U.S. Addresses

The input fields AddressLine1 through AddressLine4 are handled differently for U.S. addresses depending on whether the firm name extraction or urbanization code extraction options are enabled. If either of these options is enabled, ValidateAddress will look at the data in all four fields to validate the address and extract the requested data (firm name and/or urbanization code). If neither of these options is enabled, ValidateAddress uses only the first two non-blank address line fields in its validation attempt. The data in the other address line fields is returned in the output field AdditionalInputData. For example,

AddressLine1: A1 Calle A
AddressLine2:
AddressLine3: URB Alamar
AddressLine4: Pitney Bowes Software

In this address, if either firm name extraction or urbanization code extraction were enabled, ValidateAddress would examine all four address lines. If neither firm name extraction nor urbanization code extraction were enabled, ValidateAddress would examine AddressLine1 and AddressLine3 (the first two non-blank address lines) and attempt to validate the address using that data; the data in AddressLine4 would be returned in the output field AdditionalInputData.

Options

Output Data Options

The following table lists the options that control the type of information returned by ValidateAddress. Some of these options can be overridden for Canadian addresses. For more information, see **Canadian Address Options** on page 493.

Table 181: Output Data Options

optionName	Description
OutputRecordType	Type of output record. For more than one, provide a list. **A** Returns 1 to 4 lines of address data plus city, state, postal code, firm name, and urbanization name information. Each address line represents an actual line of the address as it would appear on an envelope. For more information, see **Output** on page 502. If ValidateAddress could validate the address, the address lines contain the standardized address. When addresses are standardized, punctuation is removed, directionals are abbreviated, street suffixes are abbreviated, and address elements are corrected. If ValidateAddress could not validate the address, the address lines contain the address as it appeared in the input ("pass through" data). Non-validated addresses are always included as pass through data in the address line fields even if you do not specify **OutputRecordType=A**. **E** Parsed address elements. Each part of the address, such as house number, street name, street suffix,

optionName	Description
	directionals, and so on is returned in a separate field. For more information, see **Parsed Address Elements Output** on page 503. Note that if you specify "E" and specify **OutputFormattedOnFail=Y**, the parsed address elements will contain the input address for addresses that could not be validated. **I** Parsed input. This option returns the input address in parsed form regardless of whether or not ValidateAddress is able to validate the address. Each part of the input address, such as house number, street name, street suffix, directionals, and so on is returned in a separate field. Parsed input (value "I") differs from the combination of **OutputRecordType=E** and **OutputFormattedOnFail=Y** in that "I" returns all input address in parsed form, not just input that could not be validated. For more information, see **Parsed Input** on page 505. **P** Postal data. Output addresses contain additional data for each validated address. For more information, see **Postal Data Output** on page 506. **Blank** Do not return any address data or postal data.
OutputFieldLevelReturnCodes	Specifies whether to include field-level result indicators. Field-level result indicators describe how ValidateAddress handled each address element. Field-level result indicators are returned in the qualifier "Result". For example, the field-level result indicator for HouseNumber is contained in **HouseNumber.Result**. For a complete listing of result indicator output fields, see **Field-Level Result Indicators** on page 511. **N** No, do not output field-level return codes (default). **Y** Yes, output field-level return codes.
OutputFormattedOnFail	Specifies whether to return a formatted address when an address cannot be validated. The address is formatted using the preferred address format for the address's country. If this option is not selected, the output address fields are blank when ValidateAddress cannot validate the address. **Note:** This option applies only to U.S. and Canadian addresses. Formatted data will not be returned for any other address. **N** No, do not format failed addresses (default). **Y** Yes, format failed addresses. Formatted addresses are returned using the format specified by the OutputRecordType option. Note that if you specify OutputRecordType=E, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed

optionName	Description
	form. If you always want the output to contain the input address in parsed form, regardless of whether or not ValidateAddress could validate the address, specify OutputRecordType=I. Formatted addresses are returned using the format specified by the Option.OutputRecordType option. Note that if you specify Option.OutputRecordType=E, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed form. If you always want the output to contain the input address in parsed form, regardless of whether or not ValidateAddress could validate the address, specify Option.OutputRecordType=I. Formatted addresses are returned using the format specified by the **Include a standard address**, **Include address line elements**, and **Include postal information** check boxes. Note that if you select **Include address line elements**, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed form. If you always want the output to contain the input address in parsed form, regardless of whether or not ValidateAddress could validate the address, select **Include standardized input address elements**. If you specify Y, you must specify "A" and/or "E" for OutputRecordType. If you specify Y, you must specify "A" and/or "E" for Option.OutputRecordType. If you check this option, you must select **Include a standard address** and/or **Include address line elements**.
OutputStreetNameAlias	For U.S. addresses only, specifies whether or not to use a street's alias in the output. A street alias is an alternate name for a street and typically applies only to a specific range of addresses on the street. If you do not allow street aliases in the output then the street's "base" name will appear in the output regardless of whether or not there is an alias for the street. The base name is the name that applies to the entire street. **N** No, do not return street name aliases in the output. **Y** Yes, return street name aliases in the output if there is an alias for the street (default).
OutputStreetNameAlias	For U.S. addresses only, specifies how to handle street name aliases used in the input. A street alias is an alternate name for a street and typically applies only to a specific range of addresses on the street. If you enable this option, street name aliases used in the input will appear in the output. If you do not enable this option,

optionName	Description
	street name aliases in the input will be converted to the base street name in the output, with the following exceptions: • If a preferred alias is used in input the preferred alias will always be used in output. • Changed aliases used in input are always converted to the base street name in output. This is one of three options that control how ValidateAddress handles street name aliases. The other two are OutputPreferredAlias and OutputAbbreviatedAlias. **Note:** If OutputAbbreviatedAlias is enabled, the abbreviated alias will always appear in the output even if you have OutputStreetNameAlias disabled. **N** No, do not return street name aliases in the output. **Y** Yes, return street name aliases in the output if the input street name is an alias (default).
OutputAddressBlocks	Specifies whether to return a formatted version of the address as it would be printed on a physical mailpiece. Each line of the address is returned in a separate address block field. There can be up to nine address block output fields: **AddressBlock1** through **AddressBlock9**. For example, this input address: AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 Results in this address block output: AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882 AddressBlock3: UNITED STATES OF AMERICA ValidateAddress formats the address into address blocks using postal authority standards. The country name is returned using the Universal Postal Union country name. Note that the option OutputCountryFormat does not affect the country name in the address block, it only affects the name returned in the **Country** output field. For addresses outside the U.S. and Canada, if ValidateAddress is unable to validate the address, no address blocks are returned. For addresses in the U.S. and Canada, address blocks are returned even if validation fails. **N** No, do not return address blocks. Default. **Y** Yes, return address blocks.

Obtaining Congressional Districts

ValidateAddress can determine the U.S. congressional district for an address.

To obtain congressional districts, `OutputRecordType` must contain P. For more information on `OutputRecordType`, see **Output Data Options** on page 475.

Table 182: Congressional District Output

columnName	Description
USCongressionalDistrict	Congressional district number. If the address is a non-state address (for example Puerto Rico or Washington D.C.) this field is blank.

Obtaining County Names

ValidateAddress can determine the county where a particular address is located and return the county name.

Note: County names are available for U.S. addresses only.

To obtain county names, `OutputRecordType` must contain P. For more information on `OutputRecordType`, see **Output Data Options** on page 475.

Table 183: County Name Output

columnName	Description
USCountyName	County name

Obtaining FIPS County Numbers

Federal Information Processing Standards (FIPS) county numbers are numbers that identify each county in a state. Note that these numbers are only unique at the state level, not the national level. For more information, see **http://www.census.gov**.

Note: FIPS county numbers are available for U.S. addresses only.

To obtain FIPS county numbers, `OutputRecordType` must contain P. For more information on `OutputRecordType`, see **Output Data Options** on page 475.

Table 184: FIPS County Number Output

columnName	Description
USFIPSCountyNumber	FIPS (Federal Information Processing Standards) county number

Obtaining Carrier Route Codes

Carrier route codes are unique identifiers assigned to each mail carrier who delivers mail, allowing unique identification of each U.S. delivery route. ValidateAddress can return the code that represents an addressee's carrier route.

Note: Carrier route codes are available for U.S. addresses only.

To obtain carrier route codes, `OutputRecordType` must contain P. For more information on `OutputRecordType`, see **Output Data Options** on page 475.

Table 185: Carrier Route Code Output

columnName	Description
USCarrierRouteCode	Carrier route code

Creating Delivery Point Barcodes

A Delivery Point Barcode (DPBC) is a POSTNET™ barcode representation of the address. It consists of 62 bars with beginning and ending frame bars and five bars each for the ZIP + 4® Code, a value calculated based on the street address number, and a correction digit. The DPBC allows automated sortation of letter mail to the carrier level in walk sequence. ValidateAddress generates the data you need to assemble a DPBC.

Note: Delivery Point Barcodes are available for U.S. addresses only. For more information on Delivery Point Barcodes, see **http://www.usps.com**.

To generate the data needed to assemble a DPBC, `OutputRecordType` must contain P. For more information on `OutputRecordType`, see **Output Data Options** on page 475.

Table 186: Delivery Point Barcode Output

columnName	Description
PostalBarCode	The delivery point portion of the delivery point barcode.
USBCCheckDigit	Check-digit portion of the 11-digit delivery point barcode.

To assemble a DPBC you concatenate the values found in the ValidateAddress output columns as follows:

PostalCode.Base + PostalCode.Addon + PostalBarcode + USBCCheckDigit

For example, if you have the following:

- **PostalCode.Base** = 49423
- **PostalCode.Addon** = 4506
- **PostalBarcode** = 29
- **USBCCheckDigit** = 2

The assembled barcode would be:

494234506292

Default Options

The following table lists the options that control the format and processing of addresses. These are called "default options" because by default the apply to all addresses. Some of these options can be overridden for Canadian addresses. For more information, see **Canadian Address Options** on page 493.

Table 187: Default Options

optionName	Description
OutputCasing	Specifies the casing of the output data. One of the following:

optionName	Description
	M The output in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345
OutputPostalCodeSeparator	Specifies whether to use separators (spaces or hyphens) in ZIP™ Codes or Canadian postal codes. For example, a ZIP + 4® Code with the separator would be 20706-1844 and without the separator it would be 207061844. A Canadian postal code with the separator would be P5E"1S7 and without the separator it would be P5E1S7. **Y** Yes, use separator (default). **N** No, do not use separator. **Note:** Spaces are used in Canadian postal codes and hyphens in U.S. ZIP + 4® Codes.
OutputMultinationalCharacters	Specifies whether or not to return multinational characters, including diacritical marks such as umlauts or accents. (Not supported for U.S. addresses). **N** No, do not use multinational characters in the output (default). Only standard ASCII characters is returned. **Y** Yes, use multinational characters in the output.
KeepMultimatch	Indicates whether or not to return multiple address for those input addresses that have more than one possible match. **Y** Yes, return multiple matches (default). **N** No, do not return multiple matches. For more information, see **Returning Multiple Matches** on page 485.
StandardAddressFormat	Specifies where to place secondary address information for U.S. addresses. Secondary address information refers to apartment numbers, suite numbers, and similar designators. For example, in this address the secondary address information is "Apt 10E" and the primary address information is "424 Washington Blvd". Apt 10E 424 Washington Blvd Springfield MI 49423 **C** Place both primary and secondary address information in AddressLine1 (default).

optionName	Description
	S Place the primary address information in AddressLine1 and the secondary address information in AddressLine2. **D** Place both primary and secondary address information in AddressLine1 and place dropped information from dual addresses in AddressLine2. A dual address is an address that contains both street information and PO Box/Rural Route/Highway Contract information. For more information, see **About Dual Address Logic** on page 484.
OutputShortCityName	Specifies how to format city names that have short city name or non-mailing city name alternatives. Applies to U.S. and Canadian addresses. **Y** Returns the USPS®-approved abbreviation for the city, if there is one. The USPS® provides abbreviations for city names that are 14 characters long or longer. City abbreviations are 13 characters or less and can be used when there is limited space on the mailing label. If there is no short city name for the city, then the full city name is returned. **N** Returns the long city name (default). **S** Returns the abbreviated city name only if an abbreviated city name is used in the input address. If the input address does not use a short city name, either the long or short city name could be returned, depending on USPS® regulations for the particular city. Select this option if you are performing a CASS™ test. **V** Output the non-mailing city name (the vanity name) if the input city name is a non-mailing city name. For example, "Hollywood" is a non-mailing city name for "Los Angeles". If you do not select this option and the input city name is a non-mailing city name the long version of the mailing city is returned.
OutputCountryFormat	Specifies the format to use for the country name returned in the **Country** output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany". **E** Use English country names (default). **S** Use Spanish country names. **F** Use French country names. **G** Use German country names. **I** Use two-letter ISO abbreviation for the countries instead of country names. **U** Use Universal Postal Union abbreviation for the countries instead of country names.

optionName	Description
HomeCountry	Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses you process are in Canada, specify Canada. ValidateAddress uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields. The valid country names are: Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent And The Grenadines, Samoa, San Marino, Sao Tome And Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe

optionName	Description
DualAddressLogic	Indicates how ValidateAddress should return a match if multiple non-blank address lines are present or multiple address types are on the same address line (U.S. addresses only). **N** (Default) USPS® CASS™ regulations determine the address returned based on the following order of priority: **1.** PO Box **2.** Firm **3.** Highrise **4.** Street **5.** Rural Route **6.** General Delivery **S** Return a street match, regardless of the address line. **P** Return a PO Box match, regardless of the address line. For more information, see **About Dual Address Logic** on page 484.

About Dual Address Logic

For U.S. addresses only, the DualAddressLogic option controls whether ValidateAddress should return a street match or a PO Box/Rural Route/Highway Contract match when the address contains both street and PO Box/Rural Route/Highway Contract information in the same address line.

Note: The DualAddressLogic option has no effect if the street information is in a different address line input field than the PO Box/Rural Route/Highway Contract information.

For example, given the following input address:

AddressLine1: 401 N Main St Apt 1 POB 1
City: Kemp
StateProvince: TX
PostalCode: 75143

ValidateAddress would return one of the following:

- If DualAddressLogic is set to either N or P, ValidateAddress returns the following:

  AddressLine1: PO Box 1
  City: Kemp
  StateProvince: TX
  PostalCode: 75143-0001

- If DualAddressLogic is set to S, ValidateAddress returns the following:

  AddressLine1: 401 N Main St Apt 1
  City: Kemp
  StateProvince: TX
  PostalCode: 75143-4806

The address data that is not used to standardize the address can be returned in one of two places:

- **AddressLine2**—The address information not used to standardize the address is returned in the **AddressLine2** field if you specify StandardAddressFormat=D. For more information on this option, see **Default Options** on page 480. For example, if you choose to return a street match for dual addresses,

AddressLine1: 401 N Main St Apt 1
AddressLine2: PO Box 1
City: Kemp
StateProvince: TX
PostalCode: 75143-0001

- **AdditionalInputData**—If you do not specify StandardAddressFormat=D then the address information not used to standardize the address is returned in the **AdditionalInputData** field. For more information on this option, see **Default Options** on page 480. For example, if you choose to return a street match for dual addresses,

  AddressLine1: 401 N Main St Apt 1
  City: Kemp
  StateProvince: TX
  PostalCode: 75143-0001
  AdditionalInputData: PO Box 1

Address information that is dropped can be retrieved by setting the StandardAddressFormat option to D. For more information, see **Default Options** on page 480 .

Returning Multiple Matches

If ValidateAddress finds multiple address in the postal database that are possible matches for the input address, you can have ValidateAddress return the possible matches. For example, the following address matches multiple addresses in the U.S. postal database:

PO BOX 1
New York, NY

Options

To return multiple matches, use the options described in the following table.

Table 188: Multiple Match Option

optionName	Description
KeepMultimatch	Indicates whether or not to return multiple address for those input addresses that have more than one possible match. **Y** Yes, return multiple matches (default). **N** No, do not return multiple matches.
MaximumResults	A number between 1 and 10 that indicates the maximum number of addresses to return. The default value is 1. **Note:** The difference between Keepmultimatch=N and KeepMultimatch=Y/MaximumResults=1 is that a multiple match will return a failure if KeepMultimatch=N, whereas a multiple match will return one record if KeepMultimatch=Y and MaximumResults=1.
OutputFieldLevelReturnCodes	To identify which output addresses are candidate addresses, you must specify a value of **Y** for OutputFieldLevelReturnCodes. When you do this, records that are candidate addresses will have one or more "M" values in the field-level result indicators.

Output

When you choose to return multiple matches, the addresses are returned in the address format you specify. For information on specifying address format, see **Output Data Options** on page 475. To identify which records are the candidate addresses, look for multiple "M" values in the field-level result indicators. For more information, see **Field-Level Result Indicators** on page 511.

U.S. Address Options

optionName	Description
PerformUSProcessing	Specifies whether to process U.S. addresses. If you enable U.S. address processing ValidateAddress will attempt to validate U.S. addresses. If you disable U.S. address processing, U.S. addresses will fail, meaning they are returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for U.S. address processing you must disable U.S. address processing in order for your jobs to complete successfully, regardless of whether or not they contain U.S. addresses. **Note:** You must have a valid license for U.S. address processing to successfully process U.S. addresses. If you enable U.S. address processing but are not licensed for this feature, or your license has expired, you will receive an error. **N** No, do not process U.S. addresses. **Y** Yes, process U.S. addresses. Default.
Database.US	Specifies which database to use for validating U.S. addresses. Only databases that have been defined in the US Database Resources panel in the Management Console are available.
PerformLOT	Enhanced Line of Travel (eLOT) processing assigns a Line of Travel sequence code to your addresses. Note that ValidateAddress does not sort into eLOT sequence but it provides data (the Line of Travel sequence code) that allows you to sort addresses into eLOT sequence. To perform eLOT processing you must have the eLOT database installed. **N** No, do not perform Line of Travel Processing. Default. **Y** Yes, perform Line of Travel processing. For a listing of the output fields returned by this option, see **Enhanced Line of Travel Output** on page 521.
PerformRDI	Residential Delivery Indicator (RDI™) processing checks if an address is a residential address (not a business address). To perform RDI™ processing, you must have the RDI™ database installed. If you enable both DPV® and RDI™ processing, RDI™ information is only returned if the address is a valid delivery point. If DPV® does not validate the address no RDI™ data is returned. **N** No, do not perform Residential Delivery Indicator processing. Default. **Y** Yes, perform Residential Delivery Indicator processing.

optionName	Description
PerformESM	Enhanced Street Matching (ESM) applies additional matching logic to correct misspelled or complex street names and obtain a match. ESM enables ValidateAddress to validate more addresses but it reduces performance. You cannot perform ESM when ASM is enabled. **N** No, do not perform enhanced street matching. Default. **Y** Yes, perform enhanced street matching.
PerformASM	All Street Matching (ASM) applies ESM processing as well as additional matching logic to correct errors in street names and obtain a match. It is effective at matching streets when the first letter of the street is incorrect. ASM provides the best address validation but reduces performance. **N** No, do not perform all street matching. **Y** Yes, perform all street matching. Default.
PerformDPV	Delivery Point Validation (DPV®) validates that a specific address exists, as opposed to validating that a specific address is within a range of valid addresses. CMRA processing checks if an address is for a mailbox rented from a private company, referred to as a Commercial Mail Receiving Agent (CMRA). To perform DPV and CMRA processing, you must have the DPV database installed. The DPV database contains both DPV and CMRA data. **N** No, do not perform Delivery Point Validation or CMRA processing. Default. **Y** Yes, perform Delivery Point Validation and CMRA processing. For a listing of the output fields returned by this option, see **DPV and CMRA Output** on page 523.
PerformLACSLink	The USPS® Locatable Address Conversion System (LACS) allows you to correct addresses that have changed as a result of a rural route address converting to street-style address, a PO Box renumbering, or a street-style address changing. When enabled, LACS^{Link} processing is attempted for addresses that could not be validated, or addresses were validated and flagged for LACS^{Link} conversion. To perform LACS^{Link} processing, you must have the LACS^{Link} database installed. **N** No, do not attempt LACS^{Link} conversion. Default. **Y** Yes, attempt LACS^{Link} conversion. For a listing of the output fields returned by this option, see **LACSLink Output** on page 522
PerformEWS	The Early Warning System (EWS) uses the USPS® EWS File to validate addresses that are not in the ZIP + 4® database. To perform EWS processing, you must have the EWS database installed. If an input address matches an address in the EWS file, ValidateAddress will return the following record-level result indicators:

optionName	Description
	• Status="F" • Status.Code="EWSFailure" • Status.Description="Address found in EWS table" **N** No, do not perform EWS processing. Default. **Y** Yes, perform EWS processing.
ExtractFirm	Specifies whether to extract the firm name from AddressLine1 through AddressLine4 and place it in the FirmName output field. This option works in cases where the input record's FirmName field is blank and there is more than one address line. **Y** Yes, extract the firm name. **N** No, do not extract the firm name. Default. To identify firm names in address lines, ValidateAddress scans the address lines for keywords and patterns that help it identify which fields are address lines and which are FirmName lines. Since this is done based on patterns, ValidateAddress may misidentify fields. The following tips can help ensure optimal firm extraction: • If possible, place the primary address elements in AddressLine1, the secondary elements in AddressLine2, Urbanization in AddressLine3, and firm in AddressLine4. If the address has no urbanization code, then place the firm name in AddressLine3 and leave AddressLine4 blank. For example, **AddressLine1:** 4200 Parliament Place **AddressLine2:** Suite 600 **AddressLine3:** Pitney Bowes Software **AddressLine4:** <blank> • When you define just two address lines, AddressLine2 is assigned to the secondary address most of the time. If you want to increase the chance that ValidateAddress will treat AddressLine2 as a firm name, put the firm name in AddressLine3 and leave AddressLine2 blank. • Numerics in a firm name (such as the "1" in "1 Stop Software") will increase the likelihood that ValidateAddress will treat the field as an address line. Here are some examples of firm name extraction: • In this example, AddressLine2 would get extracted into the FirmName output field **FirmName:** <blank> **AddressLine1:** 4200 Parliament Place Suite 600 **AddressLine2:** International Goose Feathers inc. • In this example, AddressLine3 would get extracted into the FirmName output field. **FirmName:** <blank> **AddressLine1:** 4200 Parliament Place **AddressLine2:** Suite 600 **AddressLine3:** Pitney Bowes Software

optionName	Description
	• In this example, AddressLine3 would be placed in the AdditionalInputData output field. The firm name would not be extracted because the FirmName input field is not blank. **FirmName:** International Goose Feathers Inc. **AddressLine1:** 4200 Parliament Place **AddressLine2:** Suite 600 **AddressLine3:** Pitney Bowes Software • In this example, no firm name would be extracted because there is only one non-blank address line, which is always treated as the primary address element. **FirmName:** <blank> **AddressLine1:** 4200 Parliament Place Suite 600 • In this example, AddressLine2 would be treated as a secondary address element because the numeral "1" causes ValidateAddress to treat that field as a secondary address element. **FirmName:** <blank> **AddressLine1:** 4200 Parliament Place Suite 600 **AddressLine2:** Pitney Bowes Software
ExtractUrb	Specifies whether to extract the urbanization name from AddressLine1 through AddressLine4 and place it in the USUrbanName output field. This option works in cases where the input record's USUrbanName field is blank and there is more than one address line. **Y** Yes, extract the urbanization name. **N** No, do not extract the urbanization name. Default. To identify urbanization names, ValidateAddress scans the address lines for keywords and patterns that help it identify which fields are address lines and which are urbanization name lines. Since this is done based on patterns, it is possible for ValidateAddress to incorrectly identify fields. To help ensure optimal urbanization extraction, place the primary address elements in AddressLine1, the secondary elements in AddressLine2, Urbanization in AddressLine3, and firm in AddressLine4, if possible. For example, **AddressLine1:** A1 Calle A **AddressLine2:** **AddressLine3:** URB Alamar **AddressLine4:** Pitney Bowes Software
PerformSuiteLink	Specifies whether to perform SuiteLink™ processing. SuiteLink corrects secondary address information for U.S. business addresses whose secondary address information could not be validated. If SuiteLink processing is enabled, the firm name is matched to a database of known firm names and their secondary address information. For example, Firm Name: Pitney Bowes Software Address Line 1: 4200 Parliament Place

optionName	Description
	Address Line 2: STE 1 Postal Code: 20706 In this case, Suite^{Link} processing would change the suite number to the correct suite number: Firm Name: Pitney Bowes Software Address Line 1: 4200 Parliament Place Address Line 2: **STE 600** Postal Code: 20706-1844 To perform Suite$^{Link™}$ processing, you must have the Suite$^{Link™}$ database installed. This option takes one of the following values: **N** No, do not use Suite$^{Link™}$. Default. **Y** Yes, use Suite$^{Link™}$ processing. For a listing of fields returned by this option, see **SuiteLink Output** on page 524.
OutputPreferredAlias	Specifies whether to use a street's preferred alias in the output. Street name aliases in the United States are alternative names given to sections of a street. There are four types of street name aliases: • **Preferred**—A preferred alias is the street name preferred locally. It typically applies only to a specific range of addresses on the street. • **Abbreviated**—An abbreviated alias is a variation of the street name that can be used in cases where the length of AddressLine1 is longer than 31 characters. For example, the street name 1234 BERKSHIRE VALLEY RD APT 312A could be abbreviated to 1234 BERKSHIRE VLLY RD APT 312A. • **Changed**—There has been an official street name change and the alias reflects the new name. For example if SHINGLE BROOK RD is changed to CANNING DR, then CANNING DR would be a changed alias type. • **Other**—The street alias is made up of other names for the street or common abbreviations of the street. The non-alias version of the street name is called the base street name. If the preferred alias is used in the input then the preferred alias will be the street name in the output regardless of whether you enable this option. This is one of three options that control how ValidateAddress handles street name aliases. The other two are OutputStreetNameAlias and OutputAbbreviatedAlias. In most cases, if you select both OutputPreferredAlias and OutputAbbreviatedAlias, and ValidateAddress finds both a preferred and an abbreviated alias in the postal database, the abbreviated alias will be used in the output. The exception to this rule is if the input street name is a preferred alias. In this case, the preferred alias will be used in the output. **Y** Yes, perform preferred alias processing.

optionName	Description
	N No, do not perform preferred alias processing. Default. **Note:** If the input address contains a street name alias of type "changed" the output address will always contain the base street name regardless of the options you specify.
OutputAbbreviatedAlias	Specifies whether to use a street's abbreviated alias in the output if the output address line is longer than 31 characters. This is one of three options that control how ValidateAddress handles street name aliases. The other two are OutputStreetNameAlias and OutputPreferredAlias. **Note:** If a preferred alias is specified in the input, the output street name will always be the preferred alias, even if you enable abbreviated street name alias processing. **Y** Yes, perform abbreviated alias processing. **N** No, do not perform abbreviated alias processing. Default. **Note:** If the input address contains a street name alias of type "changed" the output address will always contain the base street name regardless of the options you specify.
DPVDetermineNoStat	Determines the "no stat" status of an address. An address is considered "no stat" if it exists but cannot receive mail, and therefore is not counted as a delivery statistic on a carrier's route (hence the term "no stat"). Examples include buildings under construction or those that the letter carrier has identified as not likely to receive mail. **N** No, do not determine "no stat" status. Default. **Y** Yes, determine "no stat" status. **Note:** You must enable DPV processing to use this option. The result is returned in the DPVNoStat field. For more information see **LACSLink Output** on page 522
DPVDetermineVacancy	Determines if the location has been unoccupied for at least 90 days. **N** No, do not determine vacancy. Default. **Y** Yes, determine vacancy. **Note:** You must enable DPV processing to use this option. The result is returned in the DPVVacant field. For more information see **LACSLink Output** on page 522
ReturnVerimove	Returns VeriMove detail data in output. **N** No, do not return VeriMove detail data. Default. **Y** Yes, return VeriMove detail data.
StreetMatchingStrictness	Specifies the algorithm to use when determining if an input address matches an address in the postal database. One of the following: **E** The input street name must match the database exactly.

optionName	Description
	T The matching algorithm is "tight."
	M The matching algorithm is "medium" (default).
	L The matching algorithm is "loose."
FirmMatchingStrictness	Specifies the algorithm to use when determining if an input address matches an address in the postal database. One of the following:
	E The input firm name must match the database exactly.
	T The matching algorithm is "tight."
	M The matching algorithm is "medium" (default).
	L The matching algorithm is "loose."
DirectionalMatchingStrictness	Specifies the algorithm to use when determining if an input address matches an address in the postal database. One of the following:
	E The input directionals, such as the "N" in 123 N Main St., must match the database exactly.
	T The matching algorithm is "tight."
	M The matching algorithm is "medium". Default.
	L The matching algorithm is "loose."
DPVSuccessfulStatusCondition	Select the match condition where a DPV result does NOT cause a record to fail.
	F Full match
	P Partial match
	A Always. Default.
	Note: You must enable DPV processing to use this option.
FailOnCMRAMatch	Treat Commercial Mail Receiving Agency (CMRA) matches as failures?
	N No, do not treat CMRA matches as failures. Default.
	Y Yes, treat CMRA matches as failures.
	Note: You must enable DPV processing to use this option.
StandardAddressPMBLine	Specifies where ValidateAddress places Private Mailbox (PMB) information.
	N None. Do not include the PMB information in Standard Address output (default).
	1 Place the PMB information in AddressLine1. If you specify 1, you must set StandardAddressFormat to either C or D.
	2 Place the PMB information in AddressLine2.

CASS Certified Processing

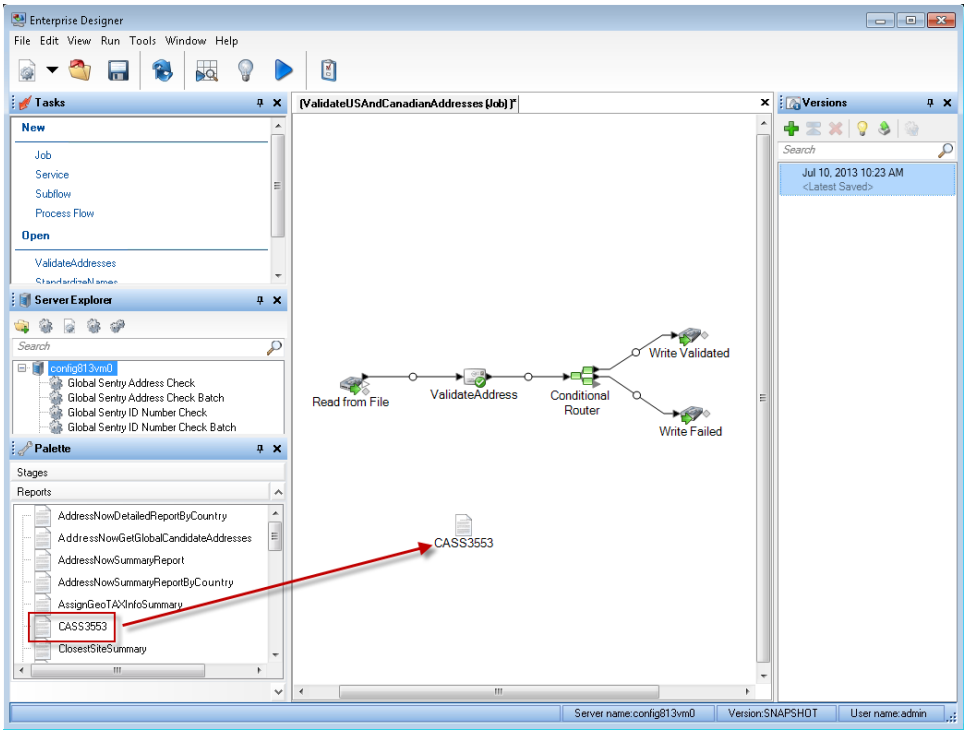
CASS Certified™ processing also generates the USPS CASS Detailed Report, which contains some of the same information as the 3553 report but provides much greater detail about DPV, LACS, and SuiteLink

statistics. The USPS CASS Detailed Report is not required for postal discounts and does not need to be submitted with your mailing.

1. Validate Address must be in CASS Certified™ mode. If **(Not CASS Certified)** appears at the top of the window, click the **Enable CASS** button. The **Enforce CASS rules** check box will appear.
2. Click **Configure CASS 3553**. The **CASS Report Fields** dialog box appears.
3. Type the **List Processor** company name, **List Name or ID#**, and the **Number of Lists** being processed for this job.
4. Type the **Mailer Name, Address**, and **City, State, ZIP**.
5. Click **OK**.

   The List information will appear in Section B and the Mailer information in Section D of the generated USPS® CASS Form 3553.

6. In Enterprise Designer, drag the **CASS3553** report from the Reports pallet to the canvas.



7. Double-click the **CASS3553** icon on the canvas.
8. On the **Stages** tab, check the **Validate Address** checkbox. Note that if you have renamed the Validate Address stage to something else, you should check the box with the name you have given the address validation stage.
9. On the **Parameters** tab, select the format for the report. You can create the report in PDF, HTML, or plain text format.
10. Click **OK**.
11. Repeat steps 6-10 for **CASSDetail** if you want to produce the CASS Detail Report.

Canadian Address Options

optionName	Description
PerformCanadianProcessing	Specifies whether to process Canadian addresses. If you enable Canadian address processing ValidateAddress will attempt to validate Canadian addresses. If you disable Canadian address processing, Canadian addresses will fail, meaning they is returned with an "F" in the Status output

optionName	Description
	field. The output field Status.Code will say "DisabledCoder." If you are not licensed for Canadian address processing you must disable Canadian address processing in order for your jobs to complete successfully, regardless of whether or not they contain Canadian addresses. **Note:** You must have a valid license for Canadian address processing to successfully process Canadian addresses. If you enable Canadian address processing but are not licensed for this feature, or your license has expired, you will receive an error. **N** No, do not process Canadian addresses. **Y** Yes, process Canadian addresses (default).
Database.Canada	Specifies which database you want to use for validating Canadian addresses. To specify a database for Canadian address validation, select a database in the **Database** drop-down list. Only databases that have been defined in the **CAN Database Resources** panel in the Management Console are available.
CanFrenchFormat	Specifies how to determine the language (English or French) to use to format the address and directional. The following example shows an address formatted in English and French: English: 123 Main St W French: 123 Rue Main O The parameter controls the formatting of the address. It also affects the spelling of the directional but not spelling of the suffix. **C** Use the street suffix returned by the matching process to determine the language. The street suffix returned by the matching process, which is used internally by ValidateAddress during processing, may be different from that in the input address. Ambiguous records are formatted like the input. Default. All addresses in Quebec are formatted using French. **S** Use the Canadian database to determine the language. The Canadian database contains data from the Canada Post Corporation (CPC). All addresses in Quebec are formatted using French. **T** Use the CanLanguage input field to determine the language. If there is a non-blank value in this field the address are formatted using French.

optionName	Description
CanEnglishApartmentLabel	For English addresses, specifies the default apartment label to use in the output if there is no apartment label in the input address. This setting is ignored if you specify CanStandardAddressFormat=F. **Apt** Use "Apt" as the label. Default. **Apartment** Use "Apartment" as the label. **Suite** Use "Suite" as the label. **Unit** Use "Unit" as the label.
CanFrenchApartmentLabel	For French addresses, specifies the default apartment label to use in the output if there is no apartment label in the input address. This setting is ignored if you specify CanStandardAddressFormat=F. **App** Use "App" as the label. Default. **Appartement** Use "Appartement" as the label. **Bureau** Use "Bureau" as the label. **Suite** Use "Suite" as the label. **Unite** Use "Unite" as the label.
ForceCorrectionLVR	Changes the civic and/or suite information to match the Large Volume Receiver (LVR) or single-single record (used when there is only one record for that postal code/street name/street type). **N** Do not change the civic and/or suite information to match the LVR or single-single record. The LVR record will be marked as a valid but non-correctable record (VN). The single-single record will be corrected, if possible, or processed as a non-correctable record.. **Y** Change the civic and/or suite information to match the LVR or single-single record. **Note:** If you check this box, the Statement of Address Accuracy will not be printed because this is **not** a SERP-recognized setting.
CanPreferHouseNum	In cases where the house number and postal code are both valid but in conflict, you can force the postal code to be corrected based on the house number by specifying CanPreferHouseNum=Y. If you do not select this option the house number is changed to match the postal code. **N** Change the house number to match the postal code. Default.

optionName	Description
	Y Change the postal code to match the house number.
CanOutputCityAlias	Specifies whether or not to return the city alias when the alias is in the input address. This option is disabled when you specify CanOutputCityFormat=D. **Y** Output the city alias when the city alias is in the input. Default. **N** Never output the city alias even if it is in the input.
CanNonCivicFormat	Specifies whether or not non-civic keywords are abbreviated in the output. For example, Post Office Box vs. PO Box. **A** Abbreviate non-civic keywords. Default. **F** Do not abbreviate non-civic keywords. The full keyword is used.
EnableSERP	Specifies whether or not to use SERP options. **Y** Enable SERP options. **N** Do not enable SERP options. Default.
CanStandardAddressFormat	Specifies where to place secondary address information in the output address. Secondary address information refers to apartment numbers, suite numbers, and similar designators. **D** Place apartment information in the location specified in the StandardAddressFormat option. Default. **B** Place apartment information at the at the end (back) of the AddressLine1 field. **F** Place the apartment number only (no label) at the beginning of the AddressLine1 field. For example, 400-123 Rue Main **E** Place the apartment number and label at the beginning of the AddressLine1 field. For example, Apt 400 123 Rue Main **S** Place apartment information on a separate line. **S** Place apartment information in the same location as the input address.
CanOutputCityFormat	Specifies whether to use the long, medium, or short version of the city if the city has a long name. For example,

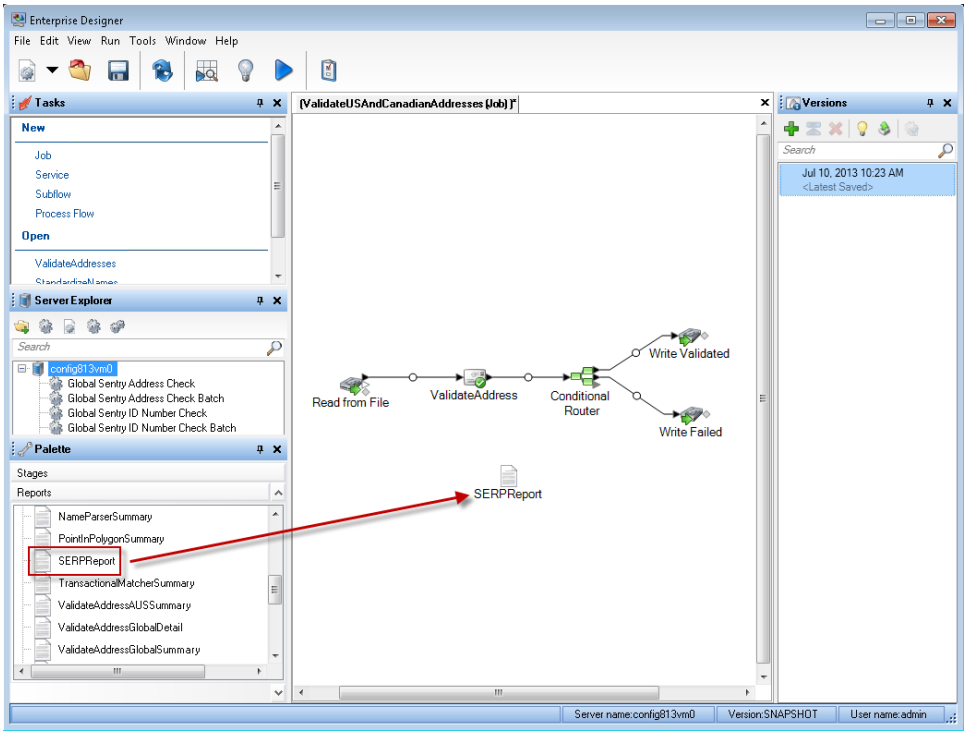
optionName	Description
	Long: BUFFALO HEAD PRAIRIE Medium: BUFFALO-HEAD-PR Short: BUFFALO-HD-PR **D** Use the default option specified by the OutputShortCityName option. Default. If you specify OutputShortCityName=V, the city is formatted as if you select `L` for this option (see below) and `Y` for **CanOutputCityAlias**. **S** Output short city name. **L** Output the long city name. **M** Output the medium city name. **I** Use the same city format as used in the input address. Output is L, M, or S.
CanRuralRouteFormat	Specifies where to place rural route delivery information. An example of an address with rural route delivery information is: 36 GRANT RD RR 3 ANTIGONISH NS In this address, "RR 3" is the rural route delivery information. **A** Place rural route delivery information on the same line as the address, after the address information. Default. For example, 36 GRANT RD RR 3 **S** Place rural route delivery information on a separate address line. For example, 36 GRANT RD RR 3
CanDeliveryOfficeFormat	Specifies where to place station information. An example of an address with station information is: PO BOX 8625 STN A ST. JOHN'S NL **I** Place station information in the same location as it is in the input address. Default. **A** Place station information on the same line as the address, after the address information. For example, PO BOX 8625 STN A

optionName	Description
	S Place station information on a separate address line. For example, PO BOX 8625 STN A
CanDualAddressLogic	Specifies whether ValidateAddress should return a street match or a PO Box/non-civic match when the address contains both civic and non-civic information. One of the following: **D** Use DualAddressLogic Global Option. Default. **P** Match to PO Box or other non-street data. **S** Match to street. For example, given the following input address: AddressLine1: 36 GRANT RD AddressLine2: RR 4 City: ANTIGONISH StateProvince: NS ValidateAddress would return one of the following: • If CanDualAddressLogic is set to S, ValidateAddress returns the following: AddressLine1: 36 GRANT RD AddressLine2: RR 3 City: ANTIGONISH StateProvince: NS PostalCode: B2G 2L1 • If CanDualAddressLogic is set to P, ValidateAddress returns the following: AddressLine1: RR 4 City: ANTIGONISH StateProvince: NS PostalCode: B2G 2L2 The address data that is not used to standardize the address is returned in the **AdditionalInputData** field. For more information on this option, see **Output Data Options** on page 475.

SERP Processing

1. Validate Address must be in SERP Certified™ mode. If **(Not SERP Certified)** appears at the top of the window, click the **Enable SERP settings** button. The **Configure SERP** box will appear.
2. Click **Configure SERP**. The **SERP Report Fields** dialog box appears.
3. Type your merchant **CPC number**.
4. Type the mailer **Name, Address**, and **City, State, ZIP**.
5. Click **OK**.

6. In Enterprise Designer, drag the SERPReport from the Reports pallet to the canvas.



7. Double-click the **SERPReport** icon on the canvas.
8. On the **Stages** tab, ensure that the **Validate Address** checkbox is checked. Note that if you have renamed the Validate Address stage to something else, you should check the box with the name you have given the address validation stage.
9. On the **Parameters** tab, select the format for the report. You can create the report in PDF, HTML, or plain text format. PDF format is the default.
10. Click **OK**.

Obtaining SERP Return Codes

SERP return codes indicate the quality of the input address as determined by the Canada Post's Software Evaluation and Recognition Program regulations.

To obtain SERP return codes, specify OutputRecordType=P. For more information on OutputRecordType, see **Output Data Options** on page 475.

SERP return codes are provided in the following output field.

Table 189: SERP Return Code Output

columnName	Description
CanadianSERPCode	Validation/correction return code (Canadian addresses only): **V** The input was valid. Canada Post defines a "valid" address as an address that meets all the following requirements: **Note:** There are exceptions. For further information, contact the CPC. • The address must contain all required components as found in CPC's Postal Code Data Files. • The address must provide an exact match on all components for only one address in CPC's Postal Code Data Files, allowing

columnName	Description
	for acceptable alternate words and names listed in the CPC Postal Code Data Files. • Address components must be in a form that allows recognition without ambiguity. Certain components may require "qualifiers" to identify them. For instance, a Route Service address requires the key words "Rural Route" or "RR" for differentiation from a "Suburban Service" or "SS" address with the same number. **I** The input was invalid. An "invalid" address is one that does not meet CPC requirements for a valid address (see above). Examples of this include address components that are missing, invalid, or inconsistent. **C** The input was correctable. A "correctable" address is one that can be corrected to match one, and only one, address. **N** The input was non-correctable. A "non-correctable" address is one that could be corrected a number of different ways such that ValidateAddress cannot identify a single correct version. **F** The input address was foreign (outside of Canada).

International Address Options

Addresses outside of the U.S. and Canada are referred to as "international" addresses. The following options control international address processing:

optionName	Description
PerformInternationalProcessing	Specifies whether to process international addresses (addresses outside the U.S. and Canada). If you enable international address processing ValidateAddress will attempt to validate international addresses. If you disable international address processing, international addresses will fail, meaning they is returned with an "F" in the Status output field. The output field Status.Code will say "DisabledCoder." If you are not licensed for international address processing you must disable international address processing in order for your jobs to complete successfully, regardless of whether or not they contain international addresses. **Note:** You must have a valid license for international address processing to successfully process international addresses. If you enable international address processing but are not licensed for this feature, or your license has expired, you will receive an error. **N** No, do not process international addresses.

optionName	Description
	Y Yes, process international addresses (default).
Database.International	Specifies which database you want to use for validating international addresses. To specify a database for international address validation, select a database in the **Database** drop-down list. Only databases that have been defined in the **INTL Database Resources** panel in the Management Console are available.
InternationalCityStreetSearching	By default, ValidateAddress provides a balance of good address matching accuracy with good performance. If you are willing to trade matching accuracy for faster performance, use the InternationalCityStreetSearching option to increase processing speed. When you do this, some accuracy is lost. This option only controls performance for addresses outside the U.S. and Canada. This setting affects a small percentage of records, mostly addresses in the U.K. There is no performance control for U.S. and Canadian address processing. If you use GetCandidateAddresses, the candidate addresses returned by GetCandidateAddresses may differ from the multiple matches returned by ValidateAddress if you set the performance tuning option for international addresses to any value other than 100. To control performance, specify a value from 0 to 100. A setting of 100 maximizes accuracy while a setting of 0 maximizes speed. The default is 100.
AddressLineSearchOnFail	This option enables ValidateAddress to search the AddressLine input fields for the city, state/province, postal code, and country when the address cannot be matched using the values in the City, StateProvince, and PostalCode input fields. Consider enabling this option if your input addresses have the city, state/province, and postal code information in the AddressLine fields. Consider disabling this option if your input addresses use the City, State/Province and PostalCode fields. If you enable this option and these fields are used, there is an increased possibility that ValidateAddress will fail to correct values in these fields (for example a misspelled city name). **N** No, do not search the AddressLine fields. **Y** Yes, search the address line fields. Default.

Output

The output from ValidateAddress contains different information depending on the output categories you select.

Standard Address Output

Standard address output consists of four lines of the address which correspond to how the address would appear on an actual address label. City, state/province, postal code, and other data is also included in standard address output. ValidateAddress returns standard address output for validated addresses if you set OutputRecordType=A. Standard address fields are always returned for addresses that could not be validated. For non-validated addresses, the standard address output fields contain the address as it appeared in the input ("pass through" data). If you want ValidateAddress to standardize address according to postal authority standards when validation fails, specify OutputFormattedOnFail=Y in your request.

Table 190: Standard Address Output

columnName	Description
AdditionalInputData	Input data not used by ValidateAddress. For more information, see **About AdditionalInputData** on page 526.
AddressLine1	If the address was validated, the first line of the validated and standardized address. If the address could not be validated, the first line of the input address without any changes.
AddressLine2	If the address was validated, the second line of the validated and standardized address. If the address could not be validated, the second line of the input address without any changes.
AddressLine3	If the address was validated, the third line of the validated and standardized address. If the address could not be validated, the third line of the input address without any changes.
AddressLine4	If the address was validated, the fourth line of the validated and standardized address. If the address could not be validated, the fourth line of the input address without any changes.
AddressLine5	For U.K. addresses only. If the address was validated, the fifth line of the validated and standardized address. If the address could not be validated, the fifth line of the input address without any changes.
City	The validated city name.
Country	The country in the format determined by what you selected in OutputCountryFormat: • ISO Code • UPU Code • English • French • German • Spanish
DepartmentName	For U.K. addresses only, a subdivision of a firm. For example, Engineering Department.

columnName	Description
FirmName	The validated firm or company name.
PostalCode	The validated ZIP Code™ or postal code.
PostalCode.AddOn	The 4-digit add-on part of the ZIP Code™. For example, in the ZIP Code™ 60655-1844, 1844 is the 4-digit add-on. (U.S. addresses only.)
PostalCode.Base	The 5-digit ZIP Code™; for example 20706 (U.S. addresses only).
StateProvince	The validated state or province abbreviation.
USUrbanName	The validated urbanization name. (U.S. addresses only.)

Parsed Address Elements Output

Output addresses are formatted in the parsed address format if you set OutputRecordType=E. If you want ValidateAddress to return formatted data in the Parsed Address format when validation fails (that is, a normalized address), specify OutputFormattedOnFail=Y.

Note: If you want ValidateAddress to always return parsed input data regardless of whether or not validation is successful, specify OutputRecordType=I. For more information, see **Parsed Input** on page 505.

Table 191: Parsed Address Output

columnName	Description
AdditionalInputData	Input data not used by ValidateAddress. For more information, see **About AdditionalInputData** on page 526.
ApartmentLabel	Apartment designator (such as STE or APT), for example: 123 E Main St **APT** 3
ApartmentLabel2	Secondary apartment designator, for example: 123 E Main St APT 3, 4th **Floor** **Note:** In this release, this field will always be blank.
ApartmentNumber	Apartment number. For example: 123 E Main St APT **3**
ApartmentNumber2	Secondary apartment number. For example: 123 E Main St APT 3, **4th** Floor **Note:** In this release, this field will always be blank.
CanadianDeliveryInstallationAreaName	Delivery installation name (Canadian addresses only)
CanadianDeliveryInstallationQualifierName	Delivery installation qualifier (Canadian addresses only)
CanadianDeliveryInstallationType	Delivery installation type (Canadian addresses only)

columnName	Description
City	Validated city name
Country	Country. Format is determined by what you selected in OutputCountryFormat: • ISO Code • UPU Code • English • French • German • Spanish
DepartmentName	For U.K. addresses only, a subdivision of a firm. For example, Engineering Department.
FirmName	The validated firm or company name
HouseNumber	House number, for example: **123** E Main St Apt 3
LeadingDirectional	Leading directional, for example: 123 **E** Main St Apt 3
POBox	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode	Validated postal code. For U.S. addresses, this is the ZIP Code.
PrivateMailbox	Private mailbox indicator.
PrivateMailbox.Type	The type of private mailbox. Possible values include: • Standard • Non-Standard **Note:** This replaces PrivateMailboxType (no period in field name). Please modify your API calls accordingly.
RRHC	Rural Route/Highway Contract indicator
StateProvince	Validated state or province name
StreetName	Street name, for example: 123 E **Main** St Apt 3
StreetSuffix	Street suffix, for example: 123 E Main **St** Apt 3
TrailingDirectional	Trailing directional, for example: 123 Pennsylvania Ave **NW**
USUrbanName	USPS® urbanization name. Puerto Rican addresses only.

Parsed Input

The output can include the input address in parsed form. This type of output is referred to as "parsed input." Parsed input fields contain the address data that was used as input regardless of whether or not ValidateAddress validated the address. Parsed input is different from the "parsed address elements" output in that parsed address elements contain the validated address if the address could be validated, and, optionally, the input address if the address could not be validated. Parsed input always contains the input address regardless of whether or not ValidateAddress validated the address.

To include parsed input fields in the output, set OutputRecordType=I.

Table 192: Parsed Input

columnName	Description
ApartmentLabel.Input	Apartment designator (such as STE or APT), for example: 123 E Main St **APT** 3
ApartmentNumber.Input	Apartment number, for example: 123 E Main St APT **3**
CanadianDeliveryInstallationAreaName.Input	Delivery installation name (Canadian addresses only)
CanadianDeliveryInstallationQualifierName.Input	Delivery installation qualifier (Canadian addresses only)
CanadianDeliveryInstallationType.Input	Delivery installation type (Canadian addresses only)
City.Input	Validated city name
Country.Input	Country. Format is determined by what you selected in OutputCountryFormat: • ISO Code • UPU Code • English • French • German • Spanish
FirmName.Input	The validated firm or company name
HouseNumber.Input	House number, for example: **123** E Main St Apt 3
LeadingDirectional.Input	Leading directional, for example: 123 **E** Main St Apt 3
POBox.Input	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode.Input	Validated postal code. For U.S. addresses, this is the ZIP Code.
PrivateMailbox.Input	Private mailbox indicator
PrivateMailbox.Type.Input	The type of private mailbox. Possible values include:

columnName	Description
	• Standard • Non-Standard
RRHC.Input	Rural Route/Highway Contract indicator
StateProvince.Input	Validated state or province name
StreetName.Input	Street name, for example: 123 E **Main St** Apt 3
StreetSuffix.Input	Street suffix, for example: 123 E Main St Apt 3
TrailingDirectional.Input	Trailing directional, for example: 123 Pennsylvania Ave **NW**
USUrbanName.Input	USPS® urbanization name

Postal Data Output

If OutputRecordType contains P then the following fields are returned in the output.

Table 193: Postal Data Output

columnName	Description
CanadianSERPCode	Validation/correction return code (Canadian addresses only). For more information, see **Obtaining SERP Return Codes** on page 499.
IntHexaviaCode	For addresses in France only, a numeric code that represents the street. For information about Hexavia codes, see **www.laposte.fr**.
IntINSEECode	For addresses in France only, a numeric code that represents the city. For a listing of INSEE codes, see **www.insee.fr**.
PostalBarCode	The two-digit delivery point portion of the delivery point barcode (U.S. addresses only) For more information, see **Creating Delivery Point Barcodes** on page 480.
USAltAddr	Indicates whether or not alternate address matching logic was used, and if so which logic was used (U.S. addresses only). One of the following: **null** No alternate address scheme used. **D** Delivery point alternate logic was used. **E** Enhanced highrise alternate match logic was used. **S** Small town default logic was used. **U** Unique ZIP Code logic was used.
USBCCheckDigit	Check-digit portion of the 11-digit delivery point barcode (U.S. addresses only). For more information, see **Creating Delivery Point Barcodes** on page 480.
USCarrierRouteCode	Carrier route code (U.S. addresses only). For more information, see **Obtaining Carrier Route Codes** on page 479.

columnName	Description
USCongressionalDistrict	Congressional district (U.S. addresses only). For more information, see **Obtaining Congressional Districts** on page 478.
USCountyName	County name (U.S. addresses only). For more information, see **Obtaining County Names** on page 479.
USFinanceNumber	The finance number in which the address resides (U.S. addresses only). The finance number is a number assigned by the USPS to an area that covers multiple ZIP Codes. ValidateAddress will successfully validate an address only if its finance number matches the finance number of the candidate address in the U.S. Database.
USFIPSCountyNumber	FIPS (Federal Information Processing Standards) county number (U.S. addresses only). For more information, see **Obtaining FIPS County Numbers** on page 479.
USLACS	Indicates whether or not the address is a candidate for LACS^{Link} conversion (U.S. addresses only). One of the following: **Y** Yes, the address is a candidate for LACS^{Link} processing. If LACS^{Link} is enabled, ValidateAddress will attempt to convert the address using the LACS^{Link} database. If the conversion attempt is successful, the output address is the new address obtained from the LACS^{Link} database. If the attempt is not successful, the address will not be converted. **N** No, the address is not a candidate for LACS^{Link} processing. LACS^{Link} processing may still be attempted if LACS^{Link} processing is requested, the LACS^{Link} database is installed, and one of the following is true: • The address matches to a Rural Route address and the RecordType.Default field returns a Y. • The input address could not be matched to any address in the U.S. Postal Database (Failures due to multiple matches are not LACS^{Link} candidates.)
USLastLineNumber	A six-character alphanumeric value that groups together ZIP Codes that share the same primary city. For example, addresses with the following two last lines would have the same last line number: Chantilly VA 20151 Chantilly VA 20152

Result Indicators

Result indicators provide information about the kinds of processing performed on an address. There are two types of result indicators:

- **Record-Level Result Indicators** on page 507
- **Field-Level Result Indicators** on page 511

Record-Level Result Indicators

Record-level result indicators provide data about the results of ValidateAddress processing for each record, such as the success or failure of the match attempt, which coder processed the address, and other details. The following table lists the record-level result indicators returned by ValidateAddress.

Table 194: Record Level Indicators

columnName	Description
AddressFormat	The type of address data being returned: **F** French format (for example: 123 Rue Main) **E** English format (for example: 123 Main St)
Confidence	The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct. For multiple matches, the confidence level is 0. For details about how this number is calculated, see **Introduction to the Validate Address Confidence Algorithm** on page 570.
CouldNotValidate	If no match was found, which address component could not be validated: • ApartmentNumber • HouseNumber • StreetName • PostalCode • City • Directional • StreetSuffix • Firm • POBoxNumber • RuralRoute **Note:** More than one component may be returned, in a comma-separated list.
CountryLevel	The category of address matching available. This is always "A" for U.S. and Canadian addresses. One of the following: **A** The address is in a country for which there is highly detailed postal data available. Addresses in this match level can have the following address elements validated and corrected, and added if missing from the input: • Postal code • City name • State/county name • Street address elements • Country name. **B** The address is in a country for which there is a medium level of postal data available. Addresses in this match level can have the following address elements validated and corrected, and added if missing from the input: • Postal code • City name • State/county name • Country name

columnName	Description
	C The address is in a country for which the postal data is least detailed. Addresses in this match level can have the following actions performed on them: • Validate and correct country name (cannot supply missing country name) • Validate the format of the postal code (cannot supply missing postal code or validate the code)
MatchScore	MatchScore provides an indication of the degree to which the output address is correct. It is significantly different from Confidence in that Confidence is indicates how much the input address changed to obtain a match, whereas the meaning of Match Score varies between U.S. and non-U.S. addresses. For U.S. addresses, MatchScore is a one-digit score on a scale of 0 to 9 that reflects the closeness of the street-name match (after transformations by ValidateAddress, if any). Zero indicates an exact match and 9 indicates the least likely match. If no match was found, this field is blank. For non-U.S. and non-Canadian addresses, MatchScore is a five-digit score, with a maximum value of 00999. Higher numbers indicates a closer match. This field does not apply to Canadian addresses. Note that you cannot equate match scores from U.S. addresses with those of non-U.S. addresses. For example, a match score of 4 for a U.S address does not indicate the same level of match as a 00004 for a non-U.S. address. **Note:** The Validate Address and Advanced Matching Module components both use the MatchScore field. The MatchScore field value in the output of a dataflow is determined by the last stage to modify the value before it is sent to an output stage. If you have a dataflow that contains Validate Address and Advanced Matching Module components and you want to see the MatchScore field output for each stage, use a Transformer stage to copy the MatchScore value to another field. For example, Validate Address produces an output field called MatchScore and then a Transformer stage copies the MatchScore field from Validate Address to a field called AddressMatchScore. When the matcher stage runs it populates the MatchScore field with the value from the matcher and passes through the AddressMatchScore value from Validate Address.
MultimatchCount	If multiple matches were found, indicates the number of records that are possible matches.
MultipleMatches	Indicates which address component had multiple matches, if multiple matches were found: • Firm • LeadingDirectional • PostalCode • StreetName • StreetSuffix • TrailingDirectional • Urbanization

columnName	Description
	Note: More than one component may be returned, in a comma-separated list.
ProcessedBy	Which address coder processed the address: **USA** U.S. address coder **CAN** Canadian address coder **INT** International address coder
RecordType	Type of address record, as defined by U.S. and Canadian postal authorities (supported for U.S. and Canadian addresses only): • FirmRecord • GeneralDelivery • HighRise • PostOfficeBox • RRHighwayContract • Normal
RecordType.Default	Code indicating the "default" match: **Y** The address matches a default record. **null** The address does not match a default record.
Status	Reports the success or failure of the match attempt. For multiple matches, this field is "F" for all the possible matches. **null** Success **F** Failure
Status.Code	Reason for failure, if there is one. For multiple matches, all possible matches is "MultipleMatchesFound." • DisabledCoder • InsufficientInputData • MultipleMatchesFound • UnableToValidate
Status.Description	Description of the problem, if there is one. **Possible Multiple Addresses Found** This value will appear if Status.Code=MultipleMatchesFound. **Address Not Found** This value will appear if Status.Code=UnableToValidate. **PerformUSProcessing disabled** This value will appear if Status.Code=DisabledCoder. **PerformCanadianProcessing disabled** This value will appear if Status.Code=DisabledCoder. **PerformInternationalProcessing disabled** This value will appear if Status.Code=DisabledCoder.

Field-Level Result Indicators

Field-level result indicators describe how ValidateAddress handled each address element. Field-level result indicators are returned in the qualifier "Result". For example, the field-level result indicator for HouseNumber is contained in **HouseNumber.Result**.

To enable field-level result indicators, specify **OutputFieldLevelReturnCodes=Y**. For more information on this option, see **Output Data Options** on page 475.

The following table lists the field-level result indicators. If a particular field does not apply to an address, the result indicator may be blank.

Table 195: Field-Level Result Indicators

columnName	Description
AddressRecord.Result	These result codes apply to international addresses only. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input.
ApartmentLabel.Result	**A** Appended. The field was added to a blank input field. U.S. and Canadian addresses only. **C** Corrected. U.S. and Canadian addresses only. **D** Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see **About AdditionalInputData** on page 526. **F** Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only. **R** The apartment label is required but is missing from the input address. U.S. addresses only. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. Does not apply to Canadian addresses.

columnName	Description	
	V	Validated. The data was confirmed correct and remained unchanged from input.
ApartmentNumber.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About AdditionalInputData** on page 526.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. addresses that are an EWS match will have a value of P. U.S. and Canadian addresses only.
	R	The apartment number is required but is missing from the input address. U.S. addresses only.
	S	Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
City.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	F	Hyphens missing or punctuation errors. Canadian addresses only.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output.
	R	The city is required but is missing from the input address. U.S. addresses only.

columnName	Description
	S Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses. **U** Unmatched. Does not apply to Canadian addresses. **V** Validated. The data was confirmed correct and remained unchanged from input.
Country.Result	These result codes do not apply to U.S. or Canadian addresses. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input.
FirmName.Result	**C** Corrected. U.S. addresses only. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only. **U** Unmatched. U.S. and Canadian addresses only. **V** Validated. The data was confirmed correct and remained unchanged from input. U.S. addresses only.
HouseNumber.Result	**A** Appended. The field was added to a blank input field. Canadian addresses only. **C** Corrected. Canadian addresses only. **D** Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About AdditionalInputData** on page 526. **F** Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses. **O** Out of range. Does not apply to U.S. or Canadian addresses. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.

columnName	Description
	R The house number is required but is missing from the input address. Canadian addresses only. **S** Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input.
LeadingDirectional.Result	**A** Appended. The field was added to a blank input field. U.S. and Canadian addresses only. **C** Corrected. Non-blank input was corrected to a non-blank value. U.S. addresses only. **D** Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About AdditionalInputData** on page 526. **F** Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input. Does not apply to Canadian addresses.
POBox.Result	**A** Appended. The field was added to a blank input field. Canadian addresses only. **C** Corrected. Canadian addresses only. **D** Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About AdditionalInputData** on page 526. **F** Formatted. The spacing and/or punctuation was changed to conform to postal

columnName	Description
	standards. Does not apply to U.S. or Canadian addresses. **M** Multiple matches. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only. **R** The P.O. Box number is required but is missing from the input address. U.S. addresses only. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input.
PostalCode.Result	**A** Appended. The field was added to a blank input field. U.S. and Canadian addresses only. **C** Corrected. U.S. and Canadian addresses only. **F** Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to Canadian addresses. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses. **R** The postal code is required but is missing from the input address. U.S. addresses only. **S** Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses. **U** Unmatched. For example, if the street name does not match the postal code, both StreetName.Result and PostalCode.Result will contain U.

columnName	Description
	V Validated. The data was confirmed correct and remained unchanged from input.
PostalCodeCity.Result	These result codes apply to international addresses only. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input.
PostalCode.Source	These result codes apply to U.S. addresses only. **FinanceNumber** The ZIP Code™ in the input was verified by using USPS® Finance Number groupings. **ZIPMOVE** The ZIP Code™ in the input address was corrected because the USPS® redrew ZIP Code™ boundaries and the address is now in a different ZIP Code™.
PostalCode.Type	**P** The ZIP Code™ contains only PO Box addresses. U.S. addresses only. **U** The ZIP Code™ is a unique ZIP Code™ assigned to a specific company or location. U.S. addresses only. **M** The ZIP Code™ is for military addresses. U.S. addresses only. **null** The ZIP Code™ is a standard ZIP Code™.
RRHC.Result	**C** Corrected. Canadian addresses only. **D** Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About AdditionalInputData** on page 526. **M** Multiple matches. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.

columnName	Description
	P Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only. **R** The rural route/highway contract is required but is missing from the input address. U.S. addresses only. **S** Standardized. This option includes any standard abbreviations. U.S. and Canadian addresses only. **U** Unmatched. U.S. and Canadian addresses only. **V** Validated. The data was confirmed correct and remained unchanged from input. U.S. and Canadian addresses only.
RRHC.Type	These result codes apply to U.S. addresses only. **HC** The address is a Highway Contract address. **RR** The address is a Rural Route address.
StateProvince.Result	**A** Appended. The field was added to a blank input field. U.S. and Canadian addresses only. **C** Corrected. U.S. addresses only. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. **R** The state is required but is missing from the input address. U.S. addresses only. **S** Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses. **U** Unmatched. Does not apply to Canadian addresses. **V** Validated. The data was confirmed correct and remained unchanged from input.
Street.Result	These result codes apply to international addresses only. **M** Multiple. The input address matched multiple records in the postal database, and

columnName	Description
	each matching record has a different value in this field. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. **R** Street corrected. House number is out of range. Applies to UK and Japanese records only. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input.
StreetName.AbbreviatedAlias.Result	Indicates the result of abbreviated alias processing. One of the following: **null** No abbreviated alias processing attempted. **B** The StreetName field contains the base street name. **L** The standardized address length is less than 31 characters so the StreetName field contains the base name. **N** No abbreviated alias found. **Y** An abbreviated alias was found for input address. The StreetName field contains the abbreviated alias.
StreetName.Alias.Type	This result code applies to U.S. addresses only. **Note:** In previous releases this field was named StreetName.AliasType with no "." between "Alias" and "Type." This old name is obsolete. Please update your processes to use the new name StreetName.Alias.Type. **Abbreviated** The alias is an abbreviation of the street name. For example, HARTS-NM RD is an abbreviated alias for HARTSVILLE NEW MARLBORO RD. **Changed** There has been an official street name change and the alias reflects the new name. For example if SHINGLE BROOK RD is changed to CANNING DR, then CANNING DR would be a changed alias type.

columnName	Description
	Other The street alias is made up of other names for the street or common abbreviations of the street. **Preferred** The street alias is the locally preferred alias. For example, a street is named "South Shore Dr." because it runs along the southern shore of a lake, not because it is south of a municipal demarcation line. So, "South" is not a predirectional in this case and should not be shorted to "S". So, "South Shore Dr." would be the preferred alias.
StreetName.PreferredAlias.Result	Indicates the result of preferred alias processing. One of the following: **null** No preferred alias processing attempted. **A** Preferred alias processing was not attempted because the input address matched to an alias. Preferred alias processing is only attempted for base addresses. **N** No preferred alias found. **Y** A preferred alias was found for the input address. The StreetName field contains the preferred alias.
StreetName.Result	**A** Appended. The field was added to a blank input field. Canadian addresses only. **C** Corrected. U.S. and Canadian addresses only. **D** Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About AdditionalInputData** on page 526. **F** Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses.

columnName	Description	
	S	Standardized. This option includes any standard abbreviations. U.S. and Canadian addresses only.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
StreetSuffix.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see **About AdditionalInputData** on page 526.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched. Does not apply to U.S. addresses.
	V	Validated. The data was confirmed correct and remained unchanged from input.
TrailingDirectional.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see **About AdditionalInputData** on page 526.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.

columnName	Description
	M Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only. **P** Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only. **S** Standardized. This option includes any standard abbreviations. **U** Unmatched. Does not apply to Canadian addresses. **V** Validated. The data was confirmed correct and remained unchanged from input.
USUrbanName.Result	These result codes apply to U.S. addresses only. **A** Appended. The field was added to a blank input field. **C** Corrected. **M** Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. **U** Unmatched. **V** Validated. The data was confirmed correct and remained unchanged from input.

Output from Options

ValidateAddress returns additional data depending on the options you select. For information on the output generated by each option, see the options listed in the following sections:

Enhanced Line of Travel Output

Enhanced Line of Travel processing produces the following output.

columnName	Description
USLOTCode	Line of Travel sequence code and an indicator denoting USPS® LOT sequence. This field is in the format nnnnY where: **nnnn** The four-digit LOT code. **Y** One of the following: • **A**—Ascending LOT sequence • **D**—Descending LOT sequence

columnName	Description
USLOTHex	A hexadecimal value that allows you to sort your file in ascending order only. The hexadecimal values range from 0 to FF ascending, then FF through 0 descending.
USLOTSequence	A two-byte value used for final sortation in place of the DPC add-on. It consists of an uppercase letter followed by a digit 0 through 9. Values range from A0 (99 descending) through J9 (00 descending), and K0 (00 ascending) through T9 (99 ascending).

LACS^{Link} Output

columnName	Description
USLACS	Indicates whether or not the address is a candidate for LACS^{Link} conversion (U.S. addresses only). One of the following: **Y** Yes, the address is a candidate for LACS^{Link} processing. If LACS^{Link} is enabled, ValidateAddress will attempt to convert the address using the LACS^{Link} database. If the conversion attempt is successful, the output address is the new address obtained from the LACS^{Link} database. If the attempt is not successful, the address will not be converted. **N** No, the address is not a candidate for LACS^{Link} processing. LACS^{Link} processing may still be attempted if LACS^{Link} processing is requested, the LACS^{Link} database is installed, and one of the following is true: • The address matches to a Rural Route address and the RecordType.Default field returns a Y. • The input address could not be matched to any address in the U.S. Postal Database (Failures due to multiple matches are not LACS^{Link} candidates.)
USLACS.ReturnCode	Indicates the success or failure of LACS^{Link} processing. (U.S. addresses only.) **A** LACS^{Link} processing successful. Record matched through LACS^{Link} processing. **00** LACS^{Link} processing failed. No matching record found during LACS^{Link} processing. **09** LACS^{Link} processing matched the input address to an older highrise default address. The address has been converted. Rather than provide an imprecise address, LACS^{Link} processing does not provide a new address. **14** LACS^{Link} processing failed. Match found during LACS^{Link} processing but conversion did not occur due to other USPS® regulations. **92** LACS^{Link} processing successful. Record matched through LACS^{Link} processing. Unit number dropped on input.

columnName	Description	
	null	LACS^{Link} did not process the record, or LACS^{Link} processing was not attempted.

RDI Output

columnName	Description	
RDI	Return values indicating address type.	
	B	The address is a business address.
	R	The address is a residential address.
	M	The address is both a residential and a business address.
	null	Not checked because the address did not code at a ZIP + 4® level, or RDI™ was not performed.

DPV and CMRA Output

columnName	Description	
DPV	Indicates the results of Delivery Point Validation (DPV) processing.	
	Y	DPV confirmed. Mail can be delivered to the address.
	N	Mail cannot be delivered to the address.
	S	The building number was validated but the unit number could not be confirmed. A building number is the primary address number for a building. A unit number is a number of a distinct mailing address within a building such as an apartment, suite, floor, and so on. For example, in this address 424 is the building number and 12 is the unit number: 424 Washington Blvd. Apt. 12 Oak Park IL 60302 USA
	D	The building number was validated but the unit number was missing from input. A building number is the primary address number for a building. A unit number is a number of a distinct mailing address within a building such as an apartment, suite, floor, and so on. For example, in this address 424 is the building number and 12 is the unit number: 424 Washington Blvd. Apt. 12 Oak Park IL 60302 USA
	M	The address matches multiple valid delivery points.
	U	The address could not be confirmed because the address did not code at the ZIP + 4® level.
	V	The address caused a false-positive violation.

columnName	Description
CMRA	Indicates if the address is a Commercial Mail Receiving Agency (CMRA) **Y** Yes, the address is a CMRA. **N** No, the address is not a CMRA. **U** Unconfirmed.
DPVFootnote	DPV footnote codes. **AA** Input address matched to the ZIP + 4® file. **A1** Input address not matched to the ZIP + 4® file. **BB** Input address matched to DPV (all components). **CC** Input address primary number matched to DPV but secondary number not match (present but not valid). **N1** Input address primary number matched to DPV but high rise address missing secondary number. **M1** Input address primary number missing. **M3** Input address primary number invalid. **P1** Input address missing PO, RR or HC Box number. **RR** Input address matched to CMRA. **R1** Input address matched to CMRA but secondary number not present.
DPVVacant	Indicates whether the building is vacant (unoccupied for 90 days). One of the following: **Y** Yes, the building is vacant. **N** No, the building is not vacant. **null** The **DPVDetermineVacancy** option was not turned on.
DPVNoStat	Indicates whether the building is a "no stat" building and therefore unable to receive mail. One of the following: **Y** Yes, the building is a "no stat" building, which means the building is not receiving mail. **N** No, the building is not a "no stat" building, which means the building does receive mail. **null** The **DPVDetermineNoStat** option was not turned on.

SuiteLink Output

columnName	Description
SuiteLinkReturnCode	Indicates whether or not ValidateAddress corrected the secondary address information (U.S. addresses only). One of the following: **A** ValidateAddress corrected the secondary address information.

columnName	Description
	00 ValidateAddress did not correct the secondary address information. **null** Suite^{Link} was not performed. **XX** Suite^{Link} processing encountered an error. For example, an error would occur if the Suite^{Link} database is expired.
SuiteLinkMatchCode	Provides additional information on the Suite^{Link} match attempt. (U.S. addresses only) **A** ValidateAddress corrected the secondary address information. **B** ValidateAddress did not correct the secondary address information. No additional detail about the match attempt is available. **C** The words in the FirmName field are all "noise" words. Noise words are defined by the USPS® and are ignored when attempting to mach the firm name. Examples of noise words are "company" and "corporation". ValidateAddress is not able to correct secondary address information for firm names that consist entirely of noise words. For example "Company and Corporation" is all noise words. **D** The address is not a high-rise default address. Suite^{Link} matching is only done for high-rise default addresses. A high-rise default is a default to use when the address does not contain valid secondary information (the apartment number or apartment type is missing). **E** Suite^{Link} processing failed because the Suite^{Link} database is expired. **null** Suite^{Link} was not performed or there was an error.
SuiteLinkFidelity	Indicates how well ValidateAddress matched the firm name to the firm names in the Suite^{Link} database. **1** The firm name matches the Suite^{Link} database exactly. **2** Good match. All words in the firm name except one matched the firm name in the Suite^{Link} database. **3** Poor match. More than one word in the firm name did not match the firm name in the Suite^{Link} database. **null** Suite^{Link} could not match the firm name, or was not performed, or there was an error.

VeriMove Output

columnName	Description
VeriMoveDataBlock	Indicates whether or not ValidateAddress should return a 250-byte field containing input data to pass to VeriMove Express. This field contains the Detail Results Indicator data required by VeriMove. For more

columnName	Description
	information about the contents of this field, see the VeriMove User's Guide. One of the following: **Y** Yes, return the field VeriMoveDataBlock.. **N** No, do not return the field VeriMoveDataBlock.

About AdditionalInputData

ValidateAddress ignores some input data during the address standardization process. This extraneous data (sometimes referred to as "dropped data") is returned in the AdditionalInputData column. Some examples of dropped data include:

- Delivery instructions (for example, "Leave at back door")
- Phone numbers (for example, "555-135-8792")
- Attention lines (for example, "Attn: John Smith")

Data such as this is generally not embedded in an address. If it is embedded, ValidateAddress can usually identify this extraneous data and return it in the AdditionalInputData column.

Note: ValidateAddress does not return dropped data from split indicia addresses. A split indicia address is one where a primary address is split between multiple address lines. For example, if the primary address is "1 Green River Valley Rd" then the following would be a split indicia version of this address: 1 Green River Valley Rd 01230

If there is more than one piece of dropped data in an address, each piece of data is separated by a semicolon and a space ("; ") for U.S. addresses and a space for addresses outside the U.S. The order of dropped data in AdditionalInputData is:

1. Care of, mail stop (U.S. addresses only)
2. Other extraneous data found on address lines
3. Entire unused data lines

For example, if this is the input address:

123 Main St C/O John Smith
Apt 5 Drop at back dock
jsmith@example.com
555-123-4567
05674

Then AdditionalInputData would contain:

C/O John Smith; Apt 5 Drop At Back Dock; 555-123-4567; Jsmith@example.com; 555-123-4567

ValidateAddress can handle the following types of extraneous data:

- **Care Of Data** on page 526
- **Extraneous Data on Its Own Address Line** on page 527
- **Extraneous Data Within an Address Line** on page 527
- **Dual Addresses** on page 528

Care Of Data

For U.S. addresses only, "care of" data is returned in AdditionalInputData. The following addresses contain examples of "care of" data:

123 Main St C/O John Smith
Apt 5
05674

123 Main St
Apt 5 ATTN John Smith
05674

123 Main St Apt 5
MailStop 2
05674

Extraneous Data on Its Own Address Line

ValidateAddress returns extraneous data on its own address line for U.S. and Canadian addresses.

For U.S. addresses, ValidateAddress uses the first two non-blank address lines to perform address standardization, unless either the firm name extraction or urbanization code extraction options are enabled (see **Address Line Processing for U.S. Addresses** on page 475 for more information). Data on other address lines is returned in AdditionalInputData. In the following address, "John Smith" would be returned in AdditionalInputData because it is in the third non-blank address line and ValidateAddress only uses the first two non-blank address lines for U.S. addresses.

123 Main St
Apt 5
John Smith
05674

If one of either of the first two non-blank address lines contains extraneous data, that data is returned in AdditionalInputData. For example, in the following addresses "John Smith" would be returned in AdditionalAddressData.

123 Main St
John Smith
05674

John Smith
123 Main St
05674

In the following address both "John Smith" and "Apt 5" would both be returned in AdditionalInputData. "John Smith" would be returned because it is extraneous data in one of the first two address lines and "Apt 5" would be returned because U.S. address data must be in the first two non-blank address lines.

John Smith
123 Main St
Apt 5
05674

Extraneous Data Within an Address Line

Extraneous data that is within an address line is returned in AdditionalInputData. For example, in the following addresses "John Smith" would be returned in AdditionalInputData.

123 Main St John Smith
05674

123 Main St Apt 5 John Smith
05674

123 Main St John Smith
Apt 5
05674

123 Main St
Apt 5 John Smith
05674

For U.S. addresses, only extraneous data at the end of the address line is returned in AdditionalInputData. Extraneous data that is not at the end of an address line is not returned for U.S. addresses. For example, in the following addresses "John Smith" is not returned.

John Smith 123 Main St
05674

123 Main John Smith St
05674

The AdditionalInputData column will sometimes contain the original street name or suffix if the street name was changed to obtain a match and the street name or suffix was at the end of a line. For example this address:

Pitney Bowes Software
4200 Parlament
Lanham MD

ValidateAddress would correct the spelling of the street name and add the suffix, returning "4200 Parliament Pl" as the corrected street address and "Parlament" in AdditionalInputData.

Dual Addresses

A dual address is an address that contains both street and PO Box/Rural Route/Highway Contract information. Depending on the processing options you select, the portion of the dual address that is not used for address standardization may be returned in AdditionalInputData. For more information, see **About Dual Address Logic** on page 484.

ValidateAddressAUS

ValidateAddressAUS standardizes and validates Australian addresses using Australia Post address data. It also adds missing postal information, such as postal codes, city names, state/territory names, and more.

ValidateAddressAUS also returns result indicators about validation attempts, such as whether or not ValidateAddressAUS validated the address, and the reason for failure if the address could not be validated.

During address matching and standardization, ValidateAddressAUS separates address lines into components and compares them to the contents of a Universal Addressing Module database. If a match is found, the input address is *standardized* to the database information.

ValidateAddressAUS is part of the Universal Addressing Module.

Input

ValidateAddressAUS takes a standard address as input. All addresses use this format.

Table 196: Input Format

columnName	Format	Description
AddressLine1	String [288]	The first address line.
AddressLine2	String [288]	The second address line.
AddressLine3	String [288]	The third address line.

columnName	Format	Description
AddressLine4	String [288]	The fourth address line.
City	String [48]	The city/locality/suburb name. This can optionally be entered into one of the AddressLine fields along with the State and Postal Code.
StateProvince	String [4]	The state. This can optionally be entered into one of the AddressLine fields along with the City and Postal Code.
PostalCode	String [8]	The postal code. This can optionally be entered into one of the AddressLine fields along with the State and City.

Options

ValidateAddressAUS provides several options that enable you to control how addresses are processed and the type of information returned.

Table 197: Options

optionName	Description/Valid Values
Database	Specifies the database to be used for Australian address validation. Only databases that have been defined in the **Australia Database Resources** panel in the Management Console are available.
OutputFieldLevelReturnCodes	Outputs result fields associated with certain output elements. See **Result Codes** on page 531. Valid values are: **N** No, do not include result codes for individual fields in the output (default). **Y** Yes, include result codes for individual fields in the output.
OutputOriginalInputFields	Returns the original input data. See **Original Input Data** on page 532. Valid values are: **N** No, do not include original input data in the output (default). **Y** Yes, include original input data in the output.
OutputMatchedAddressFields	Returns parsed address elements. See **Parsed Address Elements** on page 532. Valid values are: **N** No, do not include parsed address elements in the output (default). **Y** Yes, include parsed address elements in the output.
AmasFormatting	Specifies that output address data is to be formatted using Address Matching Approval System (AMAS) conventions. This option causes Validate Address AUS to use AMAS rules when standardizing an address. AMAS is an Australia Post program for

optionName	Description/Valid Values
	enforcing addressing standards. For more information on the AMAS formatting conventions, refer to the Address Matching Approval System (AMAS) Handbook. This option modifies the output data as follows. • Numeric fields are padded with zeros. This affects the following output fields: HouseNumber, HouseNumber2, PostalDeliveryNumber, and DPID. For example, if the input address is 298 New South Head Rd Double Bay NSW 2028, then the format of the HouseNumber field is changed from 298 to 00298. • If a match is not made, then all digits in the DPID field will be zero. For example, 00000000. • If a match is not made, then all return fields (parsed address elements) will be blank, except numeric fields which will contain all zeros. • The CCD field is not output. Valid values are: **N** No, do not format the output data using AMAS conventions (default). **Y** Yes, format the output data using AMAS conventions.

Output

At a minimum, the output of ValidateAddressAUS consists of the standard output fields listed in **Standard Output Fields** on page 530. In addition to these standard fields, the output may also include other information, depending on the output options you select. For more information on the optional output fields, see **Result Codes** on page 531, **Parsed Address Elements** on page 532, and **Original Input Data** on page 532.

Standard Output Fields

The following table lists the standard fields that are output by ValidateAddressAUS.

Table 198: Output Fields

columnName	Description
AddressLine1	A formatted address line.
BuildingName	The building name.
City	The city/locality/suburb name 1.
City2	The city/locality/suburb name 2 - split names e.g. VIA.
StateProvince	The state.
PostalCode	The postal code.
CCD	The Census Collection District. The basic geographic unit for collection, processing and output of census data. In general, there are about 200 to 250 households per CCD, and about 37,000 CCDs throughout Australia.

columnName	Description
DPID	The Delivery Point Identifier. An eight digit number from the Australia Post Postal Address File that uniquely identifies a mail delivery point, such as a street address.
Status	The success or failure of the match attempt. **F** Failure (no DPID or CCD found) **null** Success
Status.Code	Reason for failure, if there is one. • UnableToValidate • InsufficientInputData
Status.Description	A description of the problem, if there is one.
AMAS.ResultCode	The result code returned by the underlying engine.
AMAS.ResultMessage	Any result messages returned by the underlying engine.

Result Codes

This option outputs result fields that are associated with certain output elements, as well as a result code for each result field, if available. If a result field does not have an accompanying result code, it may indicate one of the following:

- No change was made to the parsed element
- The parsed element was standardized (e.g., 'Street' changed to 'ST')
- No data was parsed into a corresponding parsed address element

Table 199: Result Codes

columnName	Result Code	
City.Result	**C**	Corrected
HouseNumber.Result	**U**	Unmatched, missing, or ambiguous
PostalCode.Result	**C**	Corrected
PostalDelivery.Result	**C**	Corrected
	D	Dropped
	U	Unmatched
StateProvince.Result	**C**	Corrected
StreetName.Result	**C**	Corrected
	U	Unmatched, missing, or ambiguous
StreetSuffix.Result	**C**	Corrected

Parsed Address Elements

This option outputs parsed address elements.

Table 200: Parsed Address Elements

columnName	Description
ApartmentLabel	The flat or unit type (such as STE or APT), for example: 123 E Main St **Apt** 3
ApartmentNumber	The flat or unit number, for example: 123 E Main St Apt **3**
FloorLabel	The floor/level type, for example: 123 E Main St Apt 3, 4th **Floor**
FloorNumber	The floor/level number, for example: 123 E Main St Apt 3, **4th** Floor
LotNumber	The lot number, for example: Lot 7 Caldwell Hwy
PostalDeliveryLabel	The postal delivery type, for example: PO Box 42
PostalDeliveryNumber	The postal delivery number, for example: PO Box 42
PostalDeliveryPrefix	The postal delivery number prefix, for example: PO Box A42
PostalDeliverySuffix	The postal delivery number suffix, for example: PO Box 42B
HouseNumber	The house number 1, for example: 298A-1B New South Head Rd
HouseSuffix	The house number 1 suffix, for example: 298A-1B New South Head Rd
HouseNumber2	The house number 2, for example: 298A-1B New South Head Rd
HouseSuffix2	The house number 2 suffix, for example: 298A-1B New South Head Rd
StreetName	The name of street where property is located, for example: 123 E **Main** St Apt 3
StreetSuffix	The street suffix, for example: 123 E Main **St** Apt 3
TrailingDirectional	The trailing directional, for example: 123 Pennsylvania Ave **NW**

Original Input Data

This option outputs the original input data in <FieldName>.Input fields.

Table 201: Input Data

columnName	Description
AddressLine1.Input	The first address line passed on input.
AddressLine2.Input	The second address line passed on input.
AddressLine3.Input	The third address line passed on input.
AddressLine4.Input	The fourth address line passed on input.
City.Input	The city/locality/suburb name passed on input.

columnName	Description
StateProvince.Input	The state passed on input.
PostalCode.Input	The postal code passed on input.

ValidateAddressGlobal

ValidateAddressGlobal provides enhanced address standardization and validation for addresses outside the U.S. and Canada. ValidateAddressGlobal can also validate addresses in the U.S. and Canada but its strength is validation of addresses in other countries. If you process a significant number of addresses outside the U.S. and Canada, you should consider using ValidateAddressGlobal.

ValidateAddressGlobal is part of the Universal Addressing Module.

ValidateAddressGlobal performs several steps to achieve a quality address, including transliteration, parsing, validation, and formatting.

Character Set Mapping and Transliteration

ValidateAddressGlobal handles international strings and their complexities. It uses fully Unicode enabled string processing which enables the transliteration of non-roman characters into the Latin character set and mapping between different character sets.

Character set mapping and transliteration features include:

- Support for over 30 different character sets including UTF-8, ISO 8859-1, GBK, BIG5, JIS, EBCDIC
- Proper "elimination" of diacritics according to language rules
- Transliteration for various alphabets into Latin Script
- Greek (BGN/PCGN 1962, ISO 843 - 1997)
- Cyrillic (BGN/PCGN 1947, ISO 9 - 1995)
- Hebrew
- Japanese Katakana, Hiragana and Kanji
- Chinese Pinyin (Mandarin, Cantonese)
- Korean Hangul

Address Parsing, Formatting, and Standardization

Restructuring incorrectly fielded address data is a complex and difficult task especially when done for international addresses. People introduce many ambiguities as they enter address data into computer systems. Among the problems are misplaced elements (such as company or personal names in street address fields) or varying abbreviations that are not only language, but also country specific. ValidateAddressGlobal identifies address elements in address lines and assigns them to the proper fields. This is an important precursor to the actual validation. Without restructuring, "no match" situations might result.

Properly identified address elements are also important when addresses have to be truncated or shortened to fit specific field length requirements. With the proper information in the right fields, specific truncation rules can be applied.

- Parses and analyzes address lines and identifies individual address elements
- Processes over 30 different character sets
- Formats addresses according to the postal rules of the country of destination
- Standardizes address elements (such as changing AVENUE to AVE)

Global Address Validation

Address validation is the correction process where properly parsed address data is compared against reference databases supplied by postal organizations or other data providers. ValidateAddressGlobal validates individual address elements to check for correctness using sophisticated fuzzy matching technology and produces standardized and formatted output based on postal standards and user preferences. FastCompletion validation type can be used in quick address entry applications. It allows input of truncated data in several address fields and generates suggestions based on this input.

In some cases, it is not possible to fully validate an address. Here ValidateAddressGlobal has a unique deliverability assessment feature that classifies addresses according to their probable deliverability.

Input

ValidateAddressGlobal takes a standard address as input. All addresses use this format no matter what country the address is from.

Table 202: ValidateAddressGlobal Input

columnName	Format	Description
AddressLine1 through AddressLine6	String [79]	These fields contain address line data. AddressLine1 contains the first address line, AddressLine2 contains the second address line, and so forth. Note that the city, state/province, and postal code information should be placed in their respective fields, not address line fields. For example: **AddressLine1:** 17413 Blodgett Road **AddressLine2:** PO Box 123 **City:** Mount Vernon **StateProvice:** WA **PostalCode:** 97273 **Country:** USA If the input address is not already parsed into the appropriate address line and City, StateProvice, and PostalCode fields, use the UnformattedLine fields instead of the address line fields.
City	String [79]	City name
StateProvince	String [79]	State or province.
PostalCode	String [79]: 99999 ~~999999999~~ A9A9A9 A9A 9A9 9999 999	The postal code for the address. In the U.S. this is the ZIP Code®.
Contact	String [79]	The name of the addressee. For example, "Mr. Jones".
Country	String [79]	The country name. If no value is specified in the `Input.ForceCountryISO3` or

columnName	Format	Description
		`Input.DefaultCountryISO3` option, you must specify a country.
FirmName	String [79]	Company or firm name
Street	String [79]	Street
Number	Building [79]	Number
Building	String [79]	Building
SubBuilding	String [79]	SubBuilding
DeliveryService	String [79]	DeliveryService
UnformattedLine1 through UnformattedLine10	String [79]	Use these fields if the input address is completely unparsed and you want ValidateAddressGlobal to attempt to parse the address into the appropriate fields. For example: **UnformattedLine1:** 17413 Blodgett Road **UnformattedLine2:** PO Box 123 **UnformattedLine3:** Mount Vernon WA 97273 **UnformattedLine4:** USA This address would be parsed into these output fields: **AddressLine1:** 17413 Blodgett Road **AddressLine2:** PO Box 123 **City:** Mount Vernon **StateProvice:** WA **PostalCode:** 97273 **Country:** USA **Note:** If you specify input in the unformatted line fields you must specify the entire address using only unformatted line fields. Do not use other fields such as City or StateProvince in combination with unformatted line fields.

Options

Input Options

Table 203: ValidateAddressGlobal Input Options

optionName	Description/Valid Values
Database.AddressGlobal	Specifies the database resource containing the postal data to use for address validation. Only databases that have been defined in the **Global Database**

optionName	Description/Valid Values
	Resources panel in the Management Console are available. For more information, see the *Spectrum™ Technology Platform Administration Guide*.
Input.DefaultCountryISO3	Specifies a default country to use when the input record does not contain explicit country information. Specify the country using the ISO3 country code. If you do not specify a default country each input record must have the country specified in the Country input field. For a list of ISO codes see **Country ISO Codes and Module Support**.
Input.ForceCountryISO3	Causes address records to be always treated as originating from the country specified here, overriding the country in the address record and the default country. Specify the country using the ISO3 country code. For a list of ISO codes, see **Country ISO Codes and Module Support**.
Input.FormatDelimiter	Enables you to use non-standard formatting for multi-line addresses in input files. Acceptable values for this field include the following: • CRLF (default) • LF • CR • SEMICOLON (2101 MASSACHUSETTS AVE NW **;** WASHINGTON DC 20008) • COMMA (2101 MASSACHUSETTS AVE NW **,** WASHINGTON DC 20008) • TAB (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) • PIPE (2101 MASSACHUSETTS AVE NW **|** WASHINGTON DC 20008) • SPACE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) **Note:** The same value must be selected for both the input option and output option.

Output Options

Table 204: ValidateAddressGlobal Output Options

optionName	Description
Result.MaximumResults	This option specifies the maximum number of candidate addresses to return. The default is 1. The maximum is 20. If you are using FastCompletion mode, you may want to enter a number greater than 1 to ensure you are provided with multiple options for completing a field.
Result.IncludeInputs	Specifies whether to include the input data in the output. If enabled, the output will contain fields that end with .Input containing the corresponding input field. For example, the output field AddressLine1.Input would contain the data specified in the input field AddressLine1. **TRUE** Include the input data in the output. **FALSE** Do not include the input data in the output (default).
Result.StateProvinceType	Specifies the format for the StateProvince field. One of the following.

optionName	Description	
	ABBREVIATION	Return the abbreviation for the state or province. For example, North Carolina would be returned as "NC".
	COUNTRY_STANDARD	Return either the abbreviation or the full name depending on the format used by the country's postal authority. (Default)
	EXTENDED	Return the full name of the state or province, not the abbreviation. For example "North Carolina".
Result.CountryType	Specifies the language or code to use for the country name returned by ValidateAddressGlobal.	
	ISO2	The two-character ISO code for the country
	ISO3	The three-character ISO code for the country
	ISO_NUMBER	The ISO country number
	NAME_CN	Chinese
	NAME_DA	Danish
	NAME_DE	German
	NAME_EN	English (default)
	NAME_ES	Spanish
	NAME_FI	Finnish
	NAME_FR	French
	NAME_GR	Greek
	NAME_HU	Hungarian
	NAME_IT	Italian
	NAME_JP	Japanese
	NAME_KR	Korean
	NAME_NL	Dutch
	NAME_PL	Polish
	NAME_PT	Portuguese
	NAME_RU	Russian
	NAME_SA	Sanskrit
	NAME_SE	Swedish
Result.PreferredScript	Specifies the alphabet in which the output should be returned. The alphabet in which the data is returned differs from country to country. For most countries the output will be Latin I regardless of the selected preferred language.	
	ASCII_Extended	ASCII characters with expansion of special characters (e.g. Ã– = OE)

optionName	Description
	ASCII_Simplified ASCII characters **Database** (default) Latin I or ASCII characters (as per reference database standard) **Latin** Latin I characters **Latin_Alt** Latin I characters (alternative transliteration) **Postal_Admin_Alt** Latin I or ASCII characters (local postal administration alternative) **Postal_Admin_Pref** Latin I or ASCII characters (as preferred by local postal administration) For countries that use an alphabet other than Latin I, the returned alphabet differs from country to country. For more information, see **Alphabets for Non-Latin 1 Countries** on page 539.
Result.PreferredLanguage	Specifies the language in which the output should be returned. The alphabet in which the data is returned differs from country to country, but for most countries the output will be Latin, regardless of the selected preferred language. **DATABASE** Language derived from reference data for each address. Default. **ENGLISH** English locality and state/province names output, if available.
Result.Casing	Specifies the casing of the output. **NATIVE** Output will be based on the reference database standard. **UPPER** Output will be in upper case for all countries. **LOWER** Output will be in lower case for all countries. **MIXED** Casing determined by country-specific rules. **NOCHANGE** For parse mode, returns the data the way it was entered. For validation mode, uses the casing found in the reference data and according to postal rules. Values that could not be checked against the reference data will retain their input casing.
Result.FormatDelimiter	Enables you to use non-standard formatting for multi-line addresses in the output. Acceptable values for this field include the following: • CRLF (default) • LF • CR • SEMICOLON (2101 MASSACHUSETTS AVE NW ; WASHINGTON DC 20008) • COMMA (2101 MASSACHUSETTS AVE NW , WASHINGTON DC 20008) • TAB (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008)

optionName	Description
	• PIPE (2101 MASSACHUSETTS AVE NW | WASHINGTON DC 20008) • SPACE (2101 MASSACHUSETTS AVE NW WASHINGTON DC 20008) **Note:** The same value must be selected for both the input option and output option.

Alphabets for Non-Latin 1 Countries

For countries that use an alphabet other than Latin I, the returned alphabet differs from country to country. The following table shows how the output is returned for specific countries. All countries that are not listed use the value specified in the field `Result.PreferredScript` option.

Country	Database	Postal_Admin_Pref	Postal_Admin_Alt	Latin	Latin_Alt	ASCII_Simplified	ASCII_Extended
RUS	Cyrillic	Cyrillic	Cyrillic	CYRILLIC_ISO	CYRILLIC_BGN	CYRILLIC_ISO + LATIN_SIMPLE	CYRILLIC_ISO + LATIN
JPN	Kanji	Kanji	Kana	JAPANESE	JAPANESE	JAPANESE + LATIN_SIMPLE	JAPANESE + LATIN
CHN	Hanzi	Hanzi	Hanzi	CHINESE_ MANDARIN	CHINESE_ CANTONESE	CHINESE_ MANDARIN + LATIN_SIMPLE	CHINESE_ MANDARIN + LATIN
HKG	Hanzi	Hanzi	Hanzi	CHINESE_ CANTONESE	CHINESE_ MANDARIN	CHINESE_ CANTONESE + LATIN_SIMPLE	CHINESE_ CANTONESE + LATIN
TWN	Hanzi	Hanzi	Hanzi	CHINESE_ CANTONESE	CHINESE_ MANDARIN	CHINESE_ CANTONESE + LATIN_SIMPLE	CHINESE_ CANTONESE + LATIN
GRC	Greek	Greek	Greek	GREEK_ISO	GREEK_BGN	GREEK_ISO + LATIN_SIMPLE	GREEK_ISO + LATIN
KOR	Latin	Hangul	Hanja	KOREAN	KOREAN	KOREAN + LATIN_SIMPLE	KOREAN + LATIN
ISR	Latin	Hebrew	Hebrew	HEBREW	HEBREW	HEBREW + LATIN_SIMPLE	HEBREW + LATIN
ROM	Latin-3	Latin-3	Latin-3	Latin-3	Latin-3	LATIN_SIMPLE	LATIN
POL	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
CZE	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
CRI	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
HUN	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
MDA	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN

Country	Database	Postal_Admin_Pref	Postal_Admin_Alt	Latin	Latin_Alt	ASCII_Simplified	ASCII_Extended
SVK	Latin-2	Latin-2	Latin-2	Latin-2	Latin-2	LATIN_SIMPLE	LATIN
LAT	Latin-7	Latin-7	Latin-7	Latin-7	Latin-7	LATIN_SIMPLE	LATIN

Process Options

Table 205: ValidateAddressGlobal Process Options

optionName	Description
Process.OptimizationLevel	Use this option to set the appropriate balance between processing speed and quality. One of the following: **NARROW** The parser will honor input assignment strictly, with the exception of separation of House Number from Street information. **STANDARD** The parser will separate address element more actively as follows: • Province will be separated from Locality information • PostalCode will be separated from Locality information • House Number will be separated from Street information • SubBuilding will be separated from Street information • DeliveryService will be separated from Street information • SubBuilding will be separated from Building information • Locality will be separated from PostalCode information **WIDE** Parser separation will happen similarly to Standard, but additionally up to 10 parsing candidates will be passed to validation for processing. Validation will widen its search tree and take additional reference data entries into account for matching. Please note that adjusting the optimization level might have no effect for countries that lack the postal reference data information required for the kind of separation described above. Increasing separation granularity from Narrow to Standard consumes some processing power, but the major impact on processing speed is from validation processing a larger search tree, thus increasing the number of data accesses and comparisons for the optimization level Wide, in an attempt to make the most out of the input data given.
Process.Mode	Specifies the type of processing to perform on the addresses. One of the following: **BATCH** Use this mode in batch processing environments when no human input or selection is possible. It is optimized for speed and will terminate its attempts to correct an address when ambiguous data is encountered that cannot be corrected automatically. The Batch processing mode will fall back to Parse mode when the database is missing for a specific country. **CERTIFIED** Use this mode in batch processing environments for Australian mail. Validate Address Global is certified

optionName	Description	
		by Australia Post's Address Matching Approval System (AMAS). It will standardize and validate your mail against the Postal Address File, providing postal discounts and allowing for the least amount of undeliverable pieces.
	FASTCOMPLETION	Use this mode if you want to use FastCompletion mode to enter truncated data in address fields and have Validate Address Global generate suggestions. For example, if you work in a call center or point-of-sale environment, you can enter just part of an address element and the FastCompletion feature will provide valid options for the complete element.
	INTERACTIVE	Use this mode when working in interactive environments to generate suggestions when an address input is ambiguous. This validation type is especially useful in data entry environments when capturing data from customers or prospects. It requires the input of an almost-complete address and will attempt to validate or correct the data provided. If ambiguities are detected, this validation type will generate up to 20 suggestions that can be used for pick lists. The Interactive processing mode will fall back to Parse mode when the respective database is missing for a specific country.
	PARSE	Use this mode for separating address input into tokens for subsequent processing in other systems, bypassing validation. For example, you could use this mode when address data of already high quality simply needs to be tokenized quickly for export to an external system or for use by a downstream stage.
Process.MatchingScope	Specifies how closely an address must match the reference data in order for the address to be validated. One of the following: **Note:** These settings may not have an effect for countries lacking the necessary level of detail in the postal reference data.	
	ALL	All address elements must match.
	DELIVERYPOINT_LEVEL	Validate Global Address must achieve a match on StateProvince, PostalCode, City/Locality/Suburb, street, house number, and sub building.
	STREET_LEVEL	Validate Global Address must achieve a match on StateProvince, PostalCode, City/Locality/Suburb, and street.
	LOCALITY_LEVEL	Validate Global Address must achieve a match on StateProvince, PostalCode, and City/Locality/Suburb.

Output

Address Data

Table 206: Parsed Address Elements

columnName	Description
AddressBlock1-9	The AddressBlock output fields contain a formatted version of the standardized or normalized address as it would be printed on a physical mailpiece. Validate Address Global formats the address into address blocks using postal authority standards. Each line of the address is returned in a separate address block field. There can be up to nine address block output fields: AddressBlock1 through AddressBlock9. For example, this input address: AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 Results in this address block output: AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882
AddressLine1-6	If the address was validated, the address line fields contain the validated and standardized address lines. If the address could not be validated, the address line fields contain the input address without any changes. Note that the last line of the address is contained in the LastLine field. For example: AddressLine1: 4200 PARLIAMENT PL STE 600 LastLine: LANHAM MD 20706-1882
AdministrativeDistrict	An area smaller than a state/province but larger than a city.
ApartmentLabel	The flat or unit type (such as STE or APT), for example: 123 E Main St **Apt** 3
ApartmentNumber	The flat or unit number, for example: 123 E Main St Apt **3**
BlockName	An estate or block name.
BuildingName	The name of a building, for example Sears Tower.
City	The name of the town or city. For example, **Vancouver**, BC.
City.AddInfo	Additional information about the city.
City.SortingCode	A code used by the postal authority to speed up delivery in certain countries for large localities, for example Prague or Dublin.
Contact	The name of the addressee. For example, **Mr. Jones**.
Country	The country in the language or code specified in the `Result.CountryType` option.
County	Dependent state or province information that further subdivides a state or province. An example would be a U.S. county.

columnName	Description
FirmName	The name of a company.
Floor	Information that further subdivides a building, e.g. the suite or apartment number. For example: 123 E Main St Apt 3, **4th Floor**
HouseNumber	The house number 1, for example: 298A-1B New South Head Rd
LastLine	Complete last address line (city, state/province, and postal code).
LeadingDirectional	Street directional that precedes the street name. For example, the N in 138 N Main Street.
Locality	Dependent place name that further subdivides a Locality. Examples are colonias in Mexico, Urbanisaciones in Spain.
POBox	Post Box descriptor (POBox, Postfach, Case Postale etc.) and number.
PostalCode	The postcode for the address. The format of the postcode varies by country.
PostalCode.AddOn	The second part of a postcode. For example, for Canadian addresses this will be the LDU. For U.S. addresses this is the ZIP + 4 add on. This field is not used by most countries.
PostalCode.Base	The base portion of the postcode.
Room	A room number in a building.
SecondaryStreet	The name of a secondary street or rural route.
StateProvince	The name of the state or province.
StreetName	The name of street where property is located, for example: 123 E **Main** St Apt 3
StreetSuffix	The street suffix, for example: 123 E Main **St** Apt 3
SubBuilding	A portion of a building, such as a suite. For example, Suite 102.
Suburb	Dependent place name that further subdivides a Locality. An example would be Mahalle in Turkey.
Territory	The name of a territory. Territories are larger than a state/province.
TrailingDirectional	The trailing directional, for example: 123 Pennsylvania Ave **NW**

Original Input Data

This option outputs the original input data in <FieldName>.Input fields.

Table 207: Original Input Data

columnName	Format	Description
AddressLine1.Input	String [79]	First address line
AddressLine2.Input	String [79]	Second address line

columnName	Format	Description
AddressLine3.Input	String [79]	Third address line
AddressLine4.Input	String [79]	Fourth address line
AddressLine5.Input	String [79]	Fifth address line
AddressLine6.Input	String [79]	Sixth address line
City.Input	String [79]	City name
StateProvince.Input	String [79]	State or province
PostalCode.Input	String [79]:	The postal code for the address. In the U.S. this is the ZIP Code. One of these formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999
Contact.Input	String [79]	The name of the addressee. For example, "Mr. Jones".
Country.Input	String [79]	Specify the country using the format you chose for input country format (English name, ISO code, or UPU code). For a list of valid values, see **Country ISO Codes and Module Support**.
FirmName.Input	String [79]	Company or firm name
Street.Input	String [79]	Street
Number.Input	Building [79]	Number
Building.Input	String [79]	Building
SubBuilding.Input	String [79]	SubBuilding
DeliveryService.Input	String [79]	DeliveryService

Result Codes

These output fields contain information about the result of the validation processing.

Table 208: Result Codes

columnName	Result Code
AddressType	For United States and Canada addresses only, the AddressType field indicates the type of address. One of the following: **F** The address was validated/corrected to the firm name. **B** The address was validated/corrected to the building name. **G** The address is a general delivery address. **H** The address was validated/corrected to the high-rise default. **L** The address is a large volume receiver. **M** The address is a military address. **P** The address was validated/corrected to PO box. **R** The address was validated/corrected to a rural route. **S** The address was validated/corrected to a street address. **U** The address could not be validated/corrected so the type is unknown.
Confidence	The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct.
CountOverflow	Indicates whether the number of candidate addresses exceeds the number returned. One of the following: **Yes** Yes, there are additional candidate addresses. To obtain the additional candidates, increase the `MaximumResults` value. **No** No, there are no additional candidates.
ElementInputStatus	ElementInputStatus provides per element information on the matching of input elements to reference data. The values in this field vary depending on whether you are using batch mode or parse mode. For information about the value in this field, see **Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance** on page 547.
ElementRelevance	Indicates which address elements are actually relevant from the local postal authority's point of view. For information about the value in this field, see **Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance** on page 547.
ElementResultStatus	ElementResultStatus categorizes the result in more detail than the ProcessStatus field by indicating if and how the output fields have been changed from the input fields. For information about the value in this field, see **Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance** on page 547.
MailabilityScore	An estimate of how likely it is that mail sent to the address would be successful delivered. One of the following: **5** Completely confident of deliverability **4** Almost certainly deliverable **3** Should be deliverable

columnName	Result Code	
	2	Fair chance
	1	Risky
	0	No chance
ModeUsed	Indicates the processing mode used. The processing mode is specified in the `Process.Mode` option. For a description of the modes, see **Process Options** on page 540.	
MultimatchCount	If the address was matched to multiple candidate addresses in the reference data, this field contains the number of candidate matches found.	
ProcessStatus	Provides a general description of the output quality. For a more detailed description of the output quality, see the ElementResultStatus field.	
	One of the following:	
	V4	Verified. The input data is correct. All elements were checked and input matched perfectly.
	V3	Verified. The input data is correct on input but some or all elements were standardized or the input contains outdated names or exonyms.
	V2	Verified. The input data is correct but some elements could not be verified because of incomplete reference data.
	V1	Verified. The input data is correct but the user standardization has deteriorated deliverability (wrong element user standardization - for example, postcode length chosen is too short). Not set by validation.
	C4	Corrected. All elements have been checked.
	C3	Corrected, but some elements could not be checked.
	C2	Corrected, but delivery status unclear (lack of reference data).
	C1	Corrected, but delivery status unclear because user standardization was wrong. Not set by validation.
	I4	Data could not be corrected completely, but is very likely to be deliverable. Single match (e.g. HNO is wrong but only 1 HNO is found in reference data).
	I3	Data could not be corrected completely, but is very likely to be deliverable. Multiple matches (e.g. HNO is wrong but more than 1 HNO is found in reference data).
	I2	Data could not be corrected, but there is a slim chance that the address is deliverable.
	I1	Data could not be corrected and is unlikely to be delivered.
	RA	Country recognized from the Force country Setting
	R9	Country recognized from DefaultCountryISO3 Setting
	R8	Country recognized from name without errors
	R7	Country recognized from name with errors
	R6	Country recognized from territory

columnName	Result Code	
	R5	Country recognized from province
	R4	Country recognized from major town
	R3	Country recognized from format
	R2	Country recognized from script
	R1	Country not recognized - multiple matches
	R0	Country not recognized
	S4	Parsed perfectly
	S3	Parsed with multiple results
	S2	Parsed with errors. Elements change position.
	S1	Parse Error. Input Format Mismatch.
	N1	Validation Error: No validation performed because country was not recognized.
	N2	Validation Error: No validation performed because required reference database is not available.
	N3	Validation Error: No validation performed because country could not be unlocked.
	N4	Validation Error: No validation performed because reference database is corrupt or in wrong format.
	N5	Validation Error: No validation performed because reference database is too old.
	N6	Validation Error: No validation performed because input data was insufficient.
	Q3	FastCompletion Status: Suggestions are available - complete address.
	Q2	FastCompletion Status: Suggested address is complete but combined with elements from the input (added or deleted).
	Q1	FastCompletion Status: Suggested address is not complete (enter more information).
	Q0	FastCompletion Status: Insufficient information provided to generate suggestions.
Status	Reports the success or failure of the processing attempt.	
	null	Success
	F	Failure
Status.Code	The reason for the failure, if there was one.	
Status.Description	A description of the reason for the failure, if there was one.	

Interpreting ElementInputStatus, ElementResultStatus, and ElementRelevance

The ElementInputStatus, ElementResultStatus, and ElementRelevance output fields contain a series of digits that describe the outcome of the validation operation in detail. ElementInputStatus contains some information for parsing operations.

This is what an ElementInputStatus value looks like:

4460604060000000060

This is what an ElementResultStatus value looks like:

88F0F870F00000000040

This is what an ElementRelevance value looks like:

11101010100000000000

To understand the values in these fields you need to know which element each position represents, and the meaning of the values in each position. For example, the first digit indicates the result from the PostalCode.Base output field. The position meanings are listed below.

- Position 1—PostalCode.Base
- Position 2—PostalCode.AddOn
- Position 3—City
- Position 4—Locality and Suburb
- Position 5—StateProvice
- Position 6—County
- Position 7—StreetName
- Position 8—SecondaryStreet
- Position 9—HouseNumber
- Position 10—Number level 1
- Position 11—POBox
- Position 12—Delivery service level 1
- Position 13—Building level 0
- Position 14—BuildingName
- Position 15—Sub building level 0
- Position 16—Floor and Room
- Position 17—FirmName
- Position 18—Organization level 1
- Position 19—Country
- Position 20—Territory

For ElementInputStatus, the possible values for validation are:

- 0—Empty
- 1—Not found
- 2—Not checked (no reference data)
- 3—Wrong - Set by validation only: The reference database suggests that either Number or DeliveryService is out of valid number range. Input is copied, not corrected for batch mode, for interactive mode and FastCompletion suggestions are provided.
- 4—Matched with errors in this element
- 5—Matched with changes (inserts and deletes) For example:
  - Parsing: Splitting of house number for "MainSt 1"
  - Validation: Replacing input that is an exonym or dropping superfluous fielded input that is invalid according to the country reference database
- 6—Matched without errors

For ElementInputStatus, the possible values for parsing are:

- 0—Empty
- 1—Element had to be relocated
- 2—Matched but needed to be normalized
- 3—Matched

For ElementRelevance, the possible values for parsing are:

- 0—Empty
- 1—Element had to be relocated
- 2—Matched but needed to be normalized
- 3—Matched

For ElementResultStatus, the possible values are (for all address elements apart from country):

- 0—Empty
- 1—Not validated and not changed. Original is copied.
- 2—Not validated but standardized.
- 3—Validated but not changed due to invalid input, database suggests that number is out of valid ranges. Input is copied, not corrected - this status value is only set in batch mode.
- 4—Validated but not changed due to lack of reference data.
- 5—Validated but not changed due to multiple matches. Only set in batch mode, otherwise multiple suggestions that replace the input are marked as corrected (status value 7).
- 6—Validated and changed by eliminating the input value
- 7—Validated and changed due to correction based on reference data
- 8—Validated and changed by adding value based on reference data
- 9—Validated, not changed, but delivery status not clear (e.g. DPV value wrong; given number ranges that only partially match reference data).
- C—Validated, verified but changed due to outdated name
- D—Validated, verified but changed from exonym to official name
- E—Validated, verified but changed due to standardization based on casing or language. Validation only sets this status if input fully matches a language alternative.
- F—Validated, verified and not changed due to perfect match

For Country (position 19 & 20), the following values are possible:

- 0—Empty
- 1—Country not recognized
- 4—Country recognized from DefaultCountryISO3 setting
- 5—Country not recognized - multiple matches
- 6—Country recognized from script
- 7—Country recognized from format
- 8—Country recognized from major town
- 9—Country recognized from province
- C—Country recognized from territory
- D—Country recognized from name with errors
- E—Country recognized from name without errors
- F—Country recognized from ForceCountryISO3 setting

ValidateAddressLoqate

ValidateAddressLoqate standardizes and validates addresses using postal authority address data. ValidateAddress Loqate can correct information and format the address using the format preferred by the applicable postal authority. It also adds missing postal information, such as postal codes, city names, state/province names, and so on.

ValidateAddressLoqate also returns result indicators about validation attempts, such as whether or not ValidateAddressLoqate validated the address, the level of confidence in the returned address, the reason for failure if the address could not be validated, and more.

During address matching and standardization, ValidateAddressLoqate separates address lines into components and compares them to the contents of the Universal Addressing Module databases. If a match is found, the input address is *standardized* to the database information. If no database match is

found, ValidateAddressLoqate optionally *formats* the input addresses. The formatting process attempts to structure the address lines according to the conventions of the appropriate postal authority.

ValidateAddressLoqate is part of the Universal Addressing Module.

Input

ValidateAddressLoqate takes an address as input. All addresses use this format regardless of the address's country. See **Address Line Processing for U.S. Addresses** on page 550 for important information about how address line data is processed for U.S. addresses.

Table 209: Input Format

columnName	Format	Description
AddressLine1	String	The first address line.
AddressLine2	String	The second address line.
AddressLine3	String	The third address line.
AddressLine4	String	The fourth address line.
City	String	The city name.
Country	String	The country code or name, in any of the following formats: • Two-character ISO 3116-1 Alpha-2 country code • Three-character ISO 3116-1 Alpha-3 country code • English country name See **Country ISO Codes and Module Support** for a list of ISO codes.
FirmName	String	The company or firm name.
PostalCode	String	The postal code for the address in one of the following formats: 99999 99999-9999 A9A9A9 A9A 9A9 9999 999
StateProvince	String	The state or province.

Address Line Processing for U.S. Addresses

The input fields AddressLine1 through AddressLine4 are handled differently for U.S. addresses depending on whether the firm name extraction or urbanization code extraction options are enabled. If either of these options is enabled, ValidateAddressLoqate will look at the data in all four fields to validate the address and extract the requested data (firm name and/or urbanization code). If neither of these options is enabled, ValidateAddressLoqate uses only the first two non-blank address line fields in its validation attempt. The data in the other address line fields is returned in the output field AdditionalInputData. For example,

AddressLine1: A1 Calle A
AddressLine2:
AddressLine3: URB Alamar
AddressLine4: Pitney Bowes Software

In this address, if either firm name extraction or urbanization code extraction were enabled, ValidateAddressLoqate would examine all four address lines. If neither firm name extraction nor urbanization code extraction were enabled, ValidateAddressLoqate would examine AddressLine1 and AddressLine3 (the first two non-blank address lines) and attempt to validate the address using that data; the data in AddressLine4 would be returned in the output field AdditionalInputData.

Options

The following table lists the options that control the type of information returned by ValidateAddressLoqate.

Table 210: Output Data Options

optionName	Description
Database.Loqate	Specifies which database you want to use for validating international addresses. To specify a database for international address validation, select a database in the **Database** drop-down list.
OutputFieldLevelReturnCodes	Specifies whether to include field-level result indicators. Field-level result indicators describe how ValidateAddressLoqate handled each address element. Field-level result indicators are returned in the qualifier "Result". For example, the field-level result indicator for HouseNumber is contained in **HouseNumber.Result**. For a complete listing of result indicator output fields, see **Result Indicators** on page 560. **N** No, do not output field-level return codes (default). **Y** Yes, output field-level return codes.
OutputFormattedOnFail	Specifies whether to return a formatted address when an address cannot be validated. The address is formatted using the preferred address format for the address's country. If this option is not selected, the output address fields are blank when ValidateAddressLoqate cannot validate the address. **Note:** This option applies only to U.S. and Canadian addresses. Formatted data will not be returned for any other address. **N** No, do not format failed addresses (default). **Y** Yes, format failed addresses. Formatted addresses are returned using the format specified by the**Include a standard address**, **Include address line elements**, and **Include postal information** check boxes. Note that if you select **Include address line elements**, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed form. If you always want the output to contain the input address in parsed form, regardless of whether or not ValidateAddressLoqate could validate the address, select **Include standardized input address elements**.

optionName	Description
	If you check this option, you must select **Include a standard address** and/or **Include address line elements**. Formatted addresses are returned using the format specified by the **OutputRecordType** option. Note that if you specify **OutputRecordType=E**, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed form. If you always want the output to contain the input address in parsed form, regardless of whether or not ValidateAddressLoqate could validate the address, specify **OutputRecordType=I**. If you specify Y, you must specify "A" and/or "E" for OutputRecordType. Formatted addresses are returned using the format specified by the **Option.OutputRecordType** option. Note that if you specify **Option.OutputRecordType=E**, the parsed address elements will contain the parsed, validated address for addresses that could be validated. If the address could not be validated the parsed address elements will contain the input address in parsed form. If you always want the output to contain the input address in parsed form, regardless of whether or not ValidateAddressLoqate could validate the address, specify **Option.OutputRecordType=I**.
OutputAddressBlocks	Specifies whether to return a formatted version of the address as it would be printed on a physical mailpiece. Each line of the address is returned in a separate address block field. There can be up to nine address block output fields: **AddressBlock1** through **AddressBlock9**. For example, this input address: AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 Results in this address block output: AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882 AddressBlock3: UNITED STATES OF AMERICA ValidateAddressLoqate formats the address into address blocks using postal authority standards. The country name is returned using the Universal Postal Union country name. Note that the option OutputCountryFormat does not affect the country name in the address block, it only affects the name returned in the **Country** output field. For addresses outside the U.S. and Canada, if ValidateAddressLoqate is unable to validate the address, no address blocks are returned. For addresses in the U.S. and Canada, address blocks are returned even if validation fails.

optionName	Description
	One of the following: **N** No, do not return address blocks. Default. **Y** Yes, return address blocks.
OutputCasing	Specifies the casing of the output data. One of the following: **M** The output in mixed case (default). For example: 123 Main St Mytown FL 12345 **U** The output in upper case. For example: 123 MAIN ST MYTOWN FL 12345
HomeCountry	Specifies the default country. You should specify the country where most of your addresses reside. For example, if most of the addresses you process are in Canada, specify Canada. ValidateAddressLoqate uses the country you specify to attempt validation when it cannot determine the country from the StateProvince, PostalCode, and Country address fields. The valid country names are: Afghanistan, Albania, Algeria, American Somoa, Andorra, Angola, Anguilla, Antigua And Barbuda, Argentina, Armenia, Aruba, Australia, Austria, Azerbaijan, Bahamas, Bahrain, Bangladesh, Barbados, Belarus, Belgium, Belize, Benin, Bermuda, Bhutan, Bolivia, Bosnia And Herzegovina, Botswana, Brazil, British Virgin Islands, Brunei Darussalam, Bulgaria, Burkina Faso, Burundi, Cambodia, Cameroon, Canada, Cape Verde, Cayman Islands, Central African Republic, Chad, Chile, China, Colombia, Comoros Islands, Congo, Cook Islands, Costa Rica, Cote D'Ivoire, Croatia, Cuba, Cyprus, Czech Republic, Democratic Republic Of Congo, Denmark, Djibouti, Dominica, Dominican Republic, East Timor, Ecuador, Egypt, El Salvador, Equitorial Guinea, Eritrea, Estonia, Ethiopia, Falkland Islands, Faroe Islands, Federated States Of Micronesia, Fiji, Finland, France, French Guiana, Gabon, Gambia, Germany, Ghana, Gibraltar, Greece, Greenland, Grenada, Guadeloupe, Guam, Guatemala, Guinea, Guinea Bissau, Guyana, Haiti, Holy See, Honduras, Hong Kong, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jamaica, Japan, Jordan, Kazakhstan, Kenya, Kiribati, Korea, Kuwait, Kyrgyzstan, Laos, Latvia, Lebanon, Lesotho, Liberia, Libya, Liechtenstein, Lithuania, Luxembourg, Macau, Macedonia, Madagascar, Malawi, Malaysia, Maldives, Mali, Malta, Marshall Islands, Martinique, Mauritania, Mauritius, Mayotte, Mexico, Moldova, Monaco, Mongolia, Monserrat, Morocco, Mozambique, Myanmar, Namibia, Nauru, Nepal, Netherlands Antilles, New Caledonia, New Zealand, Nicaragua, Niger, Nigeria, Niue, Norway, Oman, Pakistan, Palau, Panama, Papua New Guinea, Paraguay, Peru, Philippines, Pitcairn Islands, Poland, Portugal, Puerto Rico, Qatar, Republic Of

optionName	Description
	Georgia, Republic Of Korea, Republic Of Singapore, Reunion, Romania, Russia, Rwanda, Saint Helena, Saint Kitts And Nevis, Saint Lucia, Saint Pierre And Miquelon, Saint Vincent And The Grenadines, Samoa, San Marino, Sao Tome And Principe, Saudi Arabia, Senegal, Seychelles, Sierra Leone, Slovakia, Slovenia, Solomon Islands, Somalia, South Africa, Spain, Sri Lanka, Sudan, Surivalue, Swaziland, Sweden, Switzerland, Syria, Tahiti, Taiwan, Tajikistan, Tanzania, Thailand, The Netherlands, Togo, Tonga, Trinidad And Tobago, Tristan Da Cunha, Tunisia, Turkey, Turkmenistan, Turks And Caicos Islands, Tuvalu, Uganda, Ukraine, United Arab Emirates, United Kingdom, United States, Uruguay, Uzbekistan, Vanuatu, Venezuela, Vietnam, Virgin Islands (US), Wallis And Futuna, Yemen, Yugoslavia, Zambia, Zimbabwe
OutputCountryFormat	Specifies the format to use for the country name returned in the **Country** output field. For example, if you select English, the country name "Deutschland" would be returned as "Germany". **E** Use English country names (default). **I** Use two-letter ISO abbreviation for the countries instead of country names. **U** Use Universal Postal Union abbreviation for the countries instead of country names.
OutputScript	Specifies the alphabet or script in which the output should be returned. This option is bi-directional and generally takes place from Native to Latin and Latin to Native. **Input** Do not perform transliteration and provide output in the same script as the input (default). **Native** Output in the native script for the selected country wherever possible. **Latn** Use English values.
KeepMultimatch	Indicates whether or not to return multiple address for those input addresses that have more than one possible match. **Y** Yes, return multiple matches (default). **N** No, do not return multiple matches. For more information, see **Returning Multiple Matches** on page 555.
Fail Multiple Matches	Fails multiple addresses for those input addresses that have more than one possible match.

Returning Multiple Matches

If ValidateAddressLoqate finds multiple address in the postal database that are possible matches for the input address, you can have ValidateAddressLoqate return the possible matches. For example, the following address matches multiple addresses in the U.S. postal database:

PO BOX 1 New York, NY

Options

To return multiple matches, use the options described in the following table.

Table 211: Multiple Match Option

optionName	Description/Valid Values
KeepMultimatch	Indicates whether or not to return multiple address for those input addresses that have more than one possible match. **Y** Yes, return multiple matches (default). **N** No, do not return multiple matches.
MaximumResults	A number between 1 and 10 that indicates the maximum number of addresses to return. The default value is 1. **Note:** The difference between Keepmultimatch=N and KeepMultimatch=Y/MaximumResults=1 is that a multiple match will return a failure if KeepMultimatch=N, whereas a multiple match will return one record if KeepMultimatch=Y and MaximumResults=1.
OutputFieldLevelReturnCodes	To identify which output addresses are candidate addresses, you must specify a value of **Y** for OutputFieldLevelReturnCodes. When you do this, records that are candidate addresses will have one or more "M" values in the field-level result indicators.

Output

When you choose to return multiple matches, the addresses are returned in the address format you specify. For information on specifying address format, see **Options** on page 551. To identify which records are the candidate addresses, look for multiple "M" values in the field-level result indicators. For more information**Result Indicators** on page 560, see .

Output

The output from ValidateAddressLoqate contains various information depending on the output categories you select.

Standard Address Output

Standard address output consists of four lines of the address which correspond to how the address would appear on an actual address label. City, state/province, postal code, and other data is also included in standard address output. ValidateAddressLoqate returns standard address output for validated addresses if you set **OutputRecordTyp**e = **A**. Standard address fields are always returned for addresses that could not be validated regardless of whether or not you set **OutputRecordType** = **A**. For non-validated addresses, the standard address output fields contain the address as it appeared in the input ("pass

through" data). If you want ValidateAddressLoqate to standardize address according to postal authority standards when validation fails, specify **OutputFormattedOnFail** = Y in your request.

Table 212: Standard Address Output

columnName	Description
AdditionalInputData	Input data that could not be matched to a particular address component. For more information, see **About Additional Input Data**.
AddressLine1-4	If the address was validated, the first line of the validated and standardized address. If the address could not be validated, the first line of the input address without any changes. There can be up to four address block output fields: AddressLine1 through AddressLine4.
City	The validated city name.
Country	The country in the format determined by what you selected in OutputCountryFormat: • ISO Code • UPU Code • English
FirmName	The validated firm or company name.
PostalCode	The validated ZIP Code™ or postal code.
PostalCode.AddOn	The 4-digit add-on part of the ZIP Code™. For example, in the ZIP Code™ 60655-1844, 1844 is the 4-digit add-on.
PostalCode.Base	The 5-digit ZIP Code™; for example 20706.
StateProvince	The validated state or province abbreviation.

Parsed Address Elements Output

Output addresses are formatted in the parsed address format if you set **OutputRecordType** = **E**. If you want ValidateAddressLoqate to return formatted data in the Parsed Address format when validation fails (that is, a normalized address), specify **OutputFormattedOnFail** = Y.

Note: If you want ValidateAddressLoqate to always return parsed input data regardless of whether or not validation is successful, specify **OutputRecordType = I**. For more information, see **Parsed Input** on page 558.

Table 213: Parsed Address Output

columnName	Description
AddressBlock1-9	The AddressBlock output fields contain a formatted version of the standardized or normalized address as it would be printed on a physical mailpiece. Validate Address Global formats the address into address blocks using postal authority standards. Each line of the address is returned in a separate address block field. There can be up to nine

columnName	Description
	address block output fields: AddressBlock1 through AddressBlock9. For example, this input address: AddressLine1: 4200 Parliament Place AddressLine2: Suite 600 City: Lanham StateProvince: MD PostalCode: 20706 Results in this address block output: AddressBlock1: 4200 PARLIAMENT PL STE 600 AddressBlock2: LANHAM MD 20706-1882
ApartmentLabel	Apartment designator (such as STE or APT), for example: 123 E Main St **APT** 3
ApartmentNumber	Apartment number. For example: 123 E Main St APT **3**
ApartmentNumber2	Secondary apartment number. For example: 123 E Main St APT 3, **4th** Floor **Note:** In this release, this field will always be blank.
City	Validated city name
Country	Country. Format is determined by what you selected in OutputCountryFormat: • ISO Code • UPU Code • English
FirmName	The validated firm or company name
HouseNumber	House number, for example: **123** E Main St Apt 3
LeadingDirectional	Leading directional, for example: 123 **E** Main St Apt 3
POBox	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode	Validated postal code. For U.S. addresses, this is the ZIP Code.
StateProvince	Validated state or province name
StreetName	Street name, for example: 123 E **Main** St Apt 3
StreetSuffix	Street suffix, for example: 123 E Main **St** Apt 3
TrailingDirectional	Trailing directional, for example: 123 Pennsylvania Ave **NW**

Parsed Input

The output can include the input address in parsed form. This type of output is referred to as "parsed input." Parsed input fields contain the address data that was used as input regardless of whether or not ValidateAddress validated the address. Parsed input is different from the "parsed address elements" output in that parsed address elements contain the validated address if the address could be validated, and, optionally, the input address if the address could not be validated. Parsed input always contains the input address regardless of whether or not ValidateAddress validated the address.

To include parsed input fields in the output, set **OutputRecordType** = **I**.

Table 214: Parsed Input

columnName	Description
ApartmentLabel.Input	Apartment designator (such as STE or APT), for example: 123 E Main St **APT** 3
ApartmentNumber.Input	Apartment number, for example: 123 E Main St APT **3**
City.Input	Validated city name
Country.Input	Country. Format is determined by what you selected in OutputCountryFormat: • ISO Code • UPU Code • English
FirmName.Input	The validated firm or company name
HouseNumber.Input	House number, for example: **123** E Main St Apt 3
LeadingDirectional.Input	Leading directional, for example: 123 **E** Main St Apt 3
POBox.Input	Post office box number. If the address is a rural route address, the rural route box number will appear here.
PostalCode.Input	Validated postal code. For U.S. addresses, this is the ZIP Code.
StateProvince.Input	Validated state or province name
StreetName.Input	Street name, for example: 123 E **Main St** Apt 3
StreetSuffix.Input	Street suffix, for example: 123 E Main St Apt 3
TrailingDirectional.Input	Trailing directional, for example: 123 Pennsylvania Ave **NW**

Geocode Output

ValidateAddressLoqate returns the latitude/longitude, geocoding match code, dependent and double dependent localities, dependent thoroughfare, subadministrative and superadministrative areas, and the search distance as output. Match codes describe how well the geocoder matched the input address to a known address; they also describe the overall status of a match attempt. Search distance codes

represent how close the geocode is to the actual physical location of an address. The output returned is in the DataTable class. For information on the DataTable class, see the "API Fundamentals" section .

Table 215: Standard Address Output

columnName	Description
Geocode.MatchCode	This two-byte code reflects the status and level of geocode matching for an address. The first byte represents the geocoding status and is one of the following: **A** Multiple candidate geocodes were found to match the input address, and an average of these was returned **I** A geocode was able to be interpolated from the input addresses location in a range **P** A single geocode was found matching the input address **U** A geocode was not able to be generated for the input address The second byte represents the level of geocoding matching and is one of the following: **5** Delivery point (post box or subbuilding) **4** Premise or building **3** Thoroughfare **2** Locality **1** Administrative area **0** None
Latitude	Eight-digit number in degrees and calculated to five decimal places (in the format specified).
Longitude	Eight-digit number in degrees and calculated to five decimal places (in the format specified).

Table 216: City/Street/Postal Code Centroid Match Codes

Element	Match Code
Address Point	P4
Address Point Interpolated	I4
Street Centroid	A4/P3
Postal Code/City Centroid	A3/P2/A2

Note: Geocode.Match.Code does not return two coordinates for a street segment (such as the beginning and ending of a portion of a street). Instead, with input resulting in return codes of I3 (interpolated to thoroughfare or street level, where no input premise number was provided), the complete street is used in the computation.

columnName	Description
SearchDistance	The radius of accuracy in meters, providing an indication of the probable maximum distance between the given geocode and the actual physical location. This field is derived from and dependent upon the accuracy and coverage of the underlying reference data.

Result Indicators

Result indicators provide information about the kinds of processing performed on an address. There are two types of result indicators:

- **Record-Level Result Indicators**
- **Field-Level Result Indicators**

Record-Level Result Indicators

Record-level result indicators provide data about the results of ValidateAddressLoqate processing for each record, such as the success or failure of the match attempt, which coder processed the address, and other details. The following table lists the record-level result indicators returned by ValidateAddressLoqate.

Table 217: Record Level Indicators

columnName	Description
Confidence	The level of confidence assigned to the address being returned. Range is from zero (0) to 100; zero indicates failure, 100 indicates a very high level of confidence that the match results are correct. For multiple matches, the confidence level is 0. For details about how this number is calculated, see **Introduction to the Validate Address Confidence Algorithm** on page 570.
CouldNotValidate	If no match was found, which address component could not be validated: • ApartmentNumber • HouseNumber • StreetName • PostalCode • City • Directional • StreetSuffix • Firm • POBoxNumber **Note:** More than one component may be returned, in a comma-separated list.
MatchScore	MatchScore provides an indication of the similarity between the input data and the closest reference data match. It is significantly different from Confidence in that Confidence indicates how much the input address changed to obtain a match, whereas the meaning of Match Score varies between U.S. and non-U.S. addresses.

columnName	Description
	The int getFieldMatchscore (unit record, const char*) field is a decimal value between 0 and 100 that reflects the similarity between the identified input data and the closest reference data match. A result of 100 indicates that no changes other than alias, casing, or diacritic changes have been made to the input data. A result of 0 indicates that there is no similarity between the input data and closest reference data match. **Note:** The Validate Address Loqate and Advanced Matching Module components both use the MatchScore field. The MatchScore field value in the output of a dataflow is determined by the last stage to modify the value before it is sent to an output stage. If you have a dataflow that contains Validate Address Loqate and Advanced Matching Module components and you want to see the MatchScore field output for each stage, use a Transformer stage to copy the MatchScore value to another field. For example, Validate Address Loqate produces an output field called MatchScore and then a Transformer stage copies the MatchScore field from Validate Address Loqate to a field called AddressMatchScore. When the matcher stage runs it populates the MatchScore field with the value from the matcher and passes through the AddressMatchScore value from Validate Address Loqate.
ProcessedBy	Which address coder processed the address: **LOQATE** The Loqate coder processed the address.
Status	Reports the success or failure of the match attempt. For multiple matches, this field is "F" for all the possible matches. **null** Success **F** Failure
Status.Code	Reason for failure, if there is one. • UnableToValidate
Status.Description	Description of the problem, if there is one. **Address Not Found** This value will appear if Status.Code=UnableToValidate.

Field-Level Result Indicators

Field-level result indicators describe how ValidateAddressLoqate handled each address element. Field-level result indicators are returned in the qualifier "Result". For example, the field-level result indicator for HouseNumber is contained in **HouseNumber.Result**.

To enable field-level result indicators, specify **OutputFieldLevelReturnCodes=Y**.

The following table lists the field-level result indicators. If a particular field does not apply to an address, the result indicator may be blank.

Table 218: Field-Level Result Indicators

columnName	Description	
ApartmentLabel.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only.
	R	The apartment label is required but is missing from the input address. U.S. addresses only.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched. Does not apply to Canadian addresses.
	V	Validated. The data was confirmed correct and remained unchanged from input.
ApartmentNumber.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. addresses that are an EWS match will have a value of P. U.S. and Canadian addresses only.
	R	The apartment number is required but is missing from the input address. U.S. addresses only.
	S	Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
City.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.

columnName	Description	
	F	Hyphens missing or punctuation errors. Canadian addresses only.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output.
	R	The city is required but is missing from the input address. U.S. addresses only.
	S	Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.
	U	Unmatched. Does not apply to Canadian addresses.
	V	Validated. The data was confirmed correct and remained unchanged from input.
Country.Result	These result codes do not apply to U.S. or Canadian addresses.	
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
FirmName.Result	**C**	Corrected. U.S. addresses only.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only.
	U	Unmatched. U.S. and Canadian addresses only.
	V	Validated. The data was confirmed correct and remained unchanged from input. U.S. addresses only.
HouseNumber.Result	**A**	Appended. The field was added to a blank input field. Canadian addresses only.
	C	Corrected. Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	O	Out of range. Does not apply to U.S. or Canadian addresses.

columnName	Description	
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	R	The house number is required but is missing from the input address. Canadian addresses only.
	S	Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
LeadingDirectional.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. Non-blank input was corrected to a non-blank value. U.S. addresses only.
	D	Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input. Does not apply to Canadian addresses.
POBox.Result	**A**	Appended. The field was added to a blank input field. Canadian addresses only.
	C	Corrected. Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M	Multiple matches. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.

columnName	Description	
	R	The P.O. Box number is required but is missing from the input address. U.S. addresses only.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
PostalCode.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to Canadian addresses.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses.
	R	The postal code is required but is missing from the input address. U.S. addresses only.
	S	Standardized. This option includes any standard abbreviations. Does not apply to U.S. or Canadian addresses.
	U	Unmatched. For example, if the street name does not match the postal code, both StreetName.Result and PostalCode.Result will contain U.
	V	Validated. The data was confirmed correct and remained unchanged from input.
PostalCode.Type	**P**	The ZIP Code™ contains only PO Box addresses. U.S. addresses only.
	U	The ZIP Code™ is a unique ZIP Code™ assigned to a specific company or location. U.S. addresses only.
	M	The ZIP Code™ is for military addresses. U.S. addresses only.
	null	The ZIP Code™ is a standard ZIP Code™.
RRHC.Type	These result codes apply to U.S. addresses only.	
	HC	The address is a Highway Contract address.
	RR	The address is a Rural Route address.
StateProvince.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. addresses only.

columnName	Description	
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. Does not apply to U.S. or Canadian addresses.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. U.S. and Canadian addresses only.
	R	The state is required but is missing from the input address. U.S. addresses only.
	S	Standardized. This option includes any standard abbreviations. Does not apply to U.S. addresses.
	U	Unmatched. Does not apply to Canadian addresses.
	V	Validated. The data was confirmed correct and remained unchanged from input.
Street.Result	These result codes apply to international addresses only.	
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output.
	R	Street corrected. House number is out of range. Applies to UK and Japanese records only.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
StreetName.Result	**A**	Appended. The field was added to a blank input field. Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Does not apply to U.S. addresses.
	S	Standardized. This option includes any standard abbreviations. U.S. and Canadian addresses only.

columnName	Description	
	U	Unmatched.
	V	Validated. The data was confirmed correct and remained unchanged from input.
StreetSuffix.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched. Does not apply to U.S. addresses.
	V	Validated. The data was confirmed correct and remained unchanged from input.
TrailingDirectional.Result	**A**	Appended. The field was added to a blank input field. U.S. and Canadian addresses only.
	C	Corrected. U.S. and Canadian addresses only.
	D	Dropped. The field provided on input was removed. U.S. and Canadian addresses only. For more information, see **About Additional Input Data**.
	F	Formatted. The spacing and/or punctuation was changed to conform to postal standards. Does not apply to U.S. or Canadian addresses.
	M	Multiple. The input address matched multiple records in the postal database, and each matching record has a different value in this field. U.S. addresses only.
	P	Pass-through. The data was not used in the validation process, but it was preserved in the output. Canadian addresses only.
	S	Standardized. This option includes any standard abbreviations.
	U	Unmatched. Does not apply to Canadian addresses.
	V	Validated. The data was confirmed correct and remained unchanged from input.

Encountering False Positives

What is a False-Positive?

To prevent the generation of address lists, the DPV and LACS^{Link} databases include false-positive records. False-positive records are artificially manufactured addresses that reside in a false-positive table. For each negative response that occurs in a DPV or LACS^{Link} query, a query is made to the false-positive table. A match to this table (called a false-positive match) disables your DPV or LACS^{Link} key. In batch processing the job that contains the violation will complete successfully but you will not be able to run any subsequent jobs that use DPV or LACS^{Link} until you report the violation and obtain a key to reactivate DPV or LACS^{Link}.

Note: The term "seed record violation" is also used to refer to encountering false positive records. The two terms mean the same thing.

Reporting DPV False-Positive Violations

Spectrum™ Technology Platform indicates a false-positive match via messages in the server log.

Client/server calls throw an exception if a false-positive match occurs. When a DPV fals positive record violation occurs, the server log will say:

```
WARN [Log] Seed record violation for S<ZIP, ZIP+4, Address, Unit> ERROR
[Log] Feature Disabled: DPU: DPV Seed Record Violation. Seed Code:
S<Address, ZIP, ZIP+4, Unit>
```

Note: If a DPV false positive record is found, the process() method (COM, C++, Java, and .NET) will throw an exception that the feature DPU has been disabled. In C, the processMessage() function will return a non-zero value.

You can report the violation and obtain a restart key by following these steps.

1. In your browser, go to http://<yourserver>:<port>/<product code>/dpv.jsp. For example, http://localhost:8080/unc/dpv.jsp for the Universal Addressing Module and http://localhost:8080/geostan/dpv.jsp for the Enterprise Geocoding Module.
2. Enter the mailer's information into each field. The number in parentheses after each field name indicates the maximum length of the field.
3. Click **Submit** when you're done. A **File Download** dialog will appear.
4. Click **Save** to save the file to your computer. A **Save As** dialog will appear.
5. Specify a file name and location on your local hard drive (for example `c:\DPVSeedFile.txt`) and click **Save**.
6. Go to **www.g1.com/support** and log in.
7. Click **DPV & LACS^{Link} False Positive**.
8. Follow the on-screen instructions to attach your seed file and obtain a restart key.

DPV False Positive Header File Layout

The USPS® has determined the required layout of the DPV false-positive header file, which is currently defined as a fixed-length file containing two or more 180-byte records. The first record must always be the header record, whose layout is shown below.

Table 219: DPV False-Positive Header Record Layout

Position	Length	Description	Format
1-40	40	Mailer's company name	Alphanumeric

Position	Length	Description	Format
41-98	58	Mailer's address line	Alphanumeric
99-126	28	Mailer's city name	Alphanumeric
127-128	2	Mailer's state abbreviation	Alphabetic
129-137	9	Mailer's 9-digit ZIP Code	Numeric
138-146	9	Total Records Processed	Numeric
147-155	9	Total Records DPV Matched	Numeric
156-164	9	Percent Match Rate to DSF	Numeric
165-173	9	Percent Match Rate to ZIP + 4®	Numeric
174-178	5	Number of ZIP Codes on file	Numeric
179-180	2	Number of False-Positives	Numeric

The trailer record contains information regarding the DPV false-positive match. There must be one trailer record added to the false-positive file for every DPV false-positive match. The layout is shown below.

Table 220: DPV False-Positive Trailer Record Layout

Position	Length	Description	Format
1-2	2	Street predirectional	Alphanumeric
3-30	28	Street name	Alphanumeric
31-34	4	Street suffix abbreviation	Alphanumeric
35-36	2	Street postdirectional	Alphanumeric
37-46	10	Address primary number	Alphanumeric
47-50	4	Address secondary abbreviation	Alphanumeric
51-58	8	Address secondary number	Numeric
59-63	5	Matched ZIP Code	Numeric
64-67	4	Matched ZIP + 4®	Numeric
68-180	113	Filler	Spaces

Reporting LACS/Link False-Positive Violations

Spectrum™ Technology Platform indicates a false-positive match via messages in the server log. Batch jobs will fail if a false-positive match occurs and client/server calls will throw an exception.

Note: The term "seed record violation" is also used to refer to encountering false positive records. The two terms mean the same thing.

When a false positive record is encountered, the server log will say:

```
2005-05-06 17:05:38,978 WARN [com.g1.component.ValidateAddress] Seed record
 violation for RR 2 28562  31373
2005-05-06 17:05:38,978 ERROR [com.g1.component.ValidateAddress] Feature
Disabled: LLU: LACS Seed Record Violation. Seed Code: 28562  31373
2005-05-06 17:05:38,978 ERROR [com.g1.dcg.gateway.Gateway] Gateway
exception: com.g1.dcg.stage.StageException:
com.g1.dcg.component.ComponentException: Feature Disabled: LLU
2005-05-06 17:06:30,291 ERROR
[com.pb.spectrum.platform.server.runtime.core.license.impl.policy.Policy]
 Feature LACSLink Real- time is disabled.
```

Note: If a LACS^{Link} false positive record is found, the process() method (COM, C++, Java, and .NET) will throw an exception that the feature LLU has been disabled. In C, the processMessage() function will return a non-zero value.

1. In your browser, go to http://<ServerName>:<port>/<product code>/lacslink.jsp. For example, http://localhost:8080/unc/lacslink.jsp for the Universal Addressing Module and http://localhost:8080/geostan/lacslink.jsp for the Enterprise Geocoding Module.
2. Enter the mailer's information into each field. The number in parentheses after the field name indicates the maximum length of the field. Click **Submit** when you're done. A **File Download** dialog will appear.
3. Click **Save** to save the file to your computer. A **Save As** dialog will appear.
4. Specify a file name and location on your local hard drive (for example `c:\lacslink.txt`) and click **Save**.
5. Go to **www.g1.com/support** and log in.
6. Click **DPV & LACS^{Link} False Positive**.
7. Follow the on-screen instructions to attach your seed file and obtain a restart key.

ValidateAddress Confidence Algorithm

Introduction to the Validate Address Confidence Algorithm

ValidateAddress computes a confidence score for each validated address. This score describes how likely it is that the validated address is correct. Confidence code values range from 0 to 100, with a zero confidence level indicating no confidence and 100 indicating a very high level of confidence that the match results are correct. Confidence codes are calculated based on an algorithm that takes into account the match results for individual output fields. The output fields involved in this calculation include:

- Country
- City
- State
- PostalCode
- StreetName
- HouseNumber
- LeadingDirectional
- TrailingDirectional
- StreetSuffix
- ApartmentNumber

Each field has its own Weight in the algorithm. Additionally, for each field the match result could be labeled as Success, Failure, or Changed. ("Changed" refers to cases where the contents of the field have been corrected in order to get a match.) The match result—Success, Failure, or Changed—determines what the Factor is for that field. Thus, the calculation for the confidence code is a product of Weight by Factor as follows:

```
Confidence = (Weight * Factor) for City
  + (Weight * Factor) for Country
  + (Weight * Factor) for State
```

```
+ (Weight * Factor) for PostalCode
+ (Weight * Factor) for StreetName
+ (Weight * Factor) for HouseNumber
+ (Weight * Factor) for Directionals
+ (Weight * Factor) for Street Suffix
+ (Weight * Factor) for ApartmentNumber
```

Confidence Algorithm for U.S. and Canadian Addresses

The following table details the scoring and logic behind the ValidateAddress confidence algorithm for U.S. and Canadian addresses.

Table 221: Confidence Algorithm for U.S. and Canadian Addresses

Field	Weight/Match Score	Factor if Changed[4]	Factor If Filled[5]
Country	10	100%	0%
City	10	50%	75%
State	15	50%	75%
PostalCode	15	25%	25%
StreetName	15	50%	75%
HouseNumber	15	50%	75%
Directionals	10	50%	75%
StreetSuffix	5	50%	75%
ApartmentNumber	5	50%	75%

Confidence Algorithm for International Addresses

There are two confidence algorithms for addresses outside the U.S. and Canada, one for addresses in countries that use postal codes and one for addresses in countries that do not use postal codes.

The following table details the confidence algorithm for non-U.S. and non-Canadian addresses from countries that use postal codes.

Table 222: Confidence Algorithm for Countries With Postal Codes

Field	Weight/Match Score	Factor if Changed[6]	Factor If Filled[7]	Factor if Postal Data Unavailable
Country	11.1111111111111	100%	0%	0%

Field	Weight/Match Score	Factor if Changed[6]	Factor If Filled[7]	Factor if Postal Data Unavailable
City	11.1111111111111	50%	75%[8]	0%
State	16.6666666666667	100%	100	80%
PostalCode	16.6666666666667	100%	100%	80%
StreetName	16.6666666666667	50%	75%	50%
HouseNumber	16.6666666666667	50%	75%	50%
Directionals	0	50%	75%	0%
StreetSuffix	5.55555555555556	50%	75%	50%
ApartmentNumber	5.55555555555556	50%	75%	50%

The following table details confidence algorithm for countries that do not use postal codes.

Table 223: Confidence Algorithm for Countries Without Postal Codes

Field	Weight/Match Score	Factor if Changed[9]	Factor If Filled[10]	Factor if Postal Data Unavailable
Country	13.3333333333333	100%	0%	0%
City	13.3333333333333	50%	75%[11]	0%
State	20	100%	100	80%
StreetName	20	50%	75%	50%

[8] If the country is a Category C country, this value is 50%. Countries fall into one of these categories:

- **Category A**—Enables the validation and correction of an address's postal code, city name, state/county name, street address elements, and country name.
- **Category B**—Enables the validation and correction of an address's postal code, city name, state/county name, and country name. It does not support the validation or correction of street address elements.
- **Category C**—Enables the validation and correction of the country name, and the validation of the format of the postal code.

[11] If the country is a Category C country, this value is 50%. Countries fall into one of these categories:

- **Category A**—Enables the validation and correction of an address's postal code, city name, state/county name, street address elements, and country name.
- **Category B**—Enables the validation and correction of an address's postal code, city name, state/county name, and country name. It does not support the validation or correction of street address elements.
- **Category C**—Enables the validation and correction of the country name, and the validation of the format of the postal code.

Field	Weight/Match Score	Factor if Changed[9]	Factor If Filled[10]	Factor if Postal Data Unavailable
HouseNumber	20	50%	75%	50%
Directionals	0	50%	75%	0%
StreetSuffix	6.66666666666667	50%	75%	50%
ApartmentNumber	6.66666666666667	50%	75%	50%

The following table lists countries without postal codes.

Table 224: Countries Without Postal Codes

Afghanistan	Albania	Angola
Anguilla	Bahamas	Barbados
Belize	Benin	Bhutan
Botswana	Burkina Faso	Burundi
Cameroon	Cayman Islands	Central African Rep.
Chad	Cocos Islands	Columbia
Comoros	Congo (Dem. Rep.)	Congo (Rep.)
Cote d'Ivoire	Korea (North)	Djibouti
Dominica	Equatorial Guinea	Eritrea
Fiji	Gabon	Gambia
Ghana	Grenada	Guyana
Ireland	Jamaica	Kiribati
Libya	Malawi	Mali
Mauritania	Namibia	Nauru
Palaos	Panama	Peru
Qatar	Rwanda	Saint Lucia
Saint Vincent & Grenadines	Samoa	Sao Tome & Principe
Seychelles	Sierra Leone	Suriname
Tanzania	Timor	Togo
Tonga	Trinidad & Tobago	Tuvalu
Uganda	United Arab Emirates	Vanuatu
Yemen	Zimbabwe	

Universal Name Module

OpenNameParser

OpenNameParser breaks down personal and business names and other terms in the name data field into their component parts. These parsed name elements are then subsequently available to other automated operations such as name matching, name standardization, or multi-record name consolidation.

OpenNameParser does the following:

- Determines the type of a name in order to describe the function that the name performs. Name entity types are divided into two major groups: personal names and business names. Within each of these major groups are subgroups.
- Determines the form of a name in order to understand which syntax the parser should follow for parsing. Personal names usually take on a natural (signature) order or a reverse order. Business names are usually ordered hierarchically.
- Determines and labels the component parts of a name so that the syntactical relationship of each name part to the entire name is identified. The personal name syntax includes prefixes, first, middle, and last name parts, suffixes, and account description terms, among other personal name parts. The business name syntax includes the firm name and suffix terms.
- Parses conjoined personal and business names and either retains them as one record or splits them into multiple records. Examples of conjoined names include "Mr. and Mrs. John Smith" and "Baltimore Gas & Electric dba Constellation Energy".
- Parses output as records or as a list.
- Assigns a parsing score that reflects the degree of confidence that the parsing is correct.

Input

Table 225: Open Name Parser Input

columnName	Description
CultureCode	The culture of the input name data. The options are listed below. **Null (empty)** Global culture (default). **de** German. **es** Spanish. **ja** Japanese. **Note:** If you added your own domain using the Open Parser Domain Editor, the cultures and culture codes for that domain are also valid.
Name	The name you want to parse. This field is required.

Options

Parsing Options

The following table lists the options that control the parsing of names.

Table 226: Open Name Parser Parsing Options

optionName	Description
ParseNaturalOrderPersonalNames	Specifies whether to parse names where the is in the order Title, First Name, Middle Name, Last Name, and Suffix. **true** Parse personal names that are in natural order. **false** Do not parse names that are in natural order.
ParseReverseOrderPersonalNames	Specifies whether to parse names where the last name is specified first. **true** Parse personal names that are in reverse order. **false** Do not parse names that are in reverse order.
ParseConjoinedNames	Specifies whether to parse conjoined names. **true** Parse conjoined names. **false** Do not parse conjoined names.
SplitConjoinedNames	Specifies whether to separate names containing more than one individual into multiple records, for example, `Bill & Sally Smith`. **true** Split conjoined names. **false** Do not split conjoined names.
ParseBusinessNames	Specifies whether to parse business names. **true** Parse business names. **false** Do not parse business names.
OutputAsList	Specifies whether to return the parsed name elements in a list form. **true** Return the parsed elements in a list form. **false** Do not return the parsed elements in a list form.
ShortcutThreshold	Specifies how to balance performance versus quality. A faster performance will result in lower quality output; likewise, higher quality will result in slower performance. When this threshold is met, no other processing will be performed on the record. Specify a value from 0 to 100. The default is 100.

Cultures Options

The following table lists the options that control name cultures.

Table 227: Open Name Parser Cultures Options

optionName	Description
DefaultCulture	Specifies which culture(s) you want to include in the parsing grammar. Global Culture is the default selection. Specify cultures by specifying the two-character culture code in a comma-separated list in priority order. For example, to attempt to parse the name using the Spanish culture first then Japanese, you would specify: `es,ja,,`

Advanced Options

The following table lists the advanced options for name parsing.

Table 228: Open Name Parser Advanced Options

Option	Description
NaturalOrderPersonalNamesDomain	Specifies the domain to use when parsing natural order personal names. The valid values are the domain names defined in the Open Parser Domain Editor too in Enterprise Designer.
NaturalOrderPersonalNamesPriority	Specify a number between 1 and 5 that indicates the priority of the natural order personal names domain relative to the other domains that you are using. This determines the order in which you want the parsers to run. Results will be returned for the first domain that scores higher than the number set in the shortcut threshold option. If no domain reaches that threshold, results for the domain with the highest score are returned. If multiple domains reach the threshold at the same time, priority goes to the domain that was run first (determined by the order set here) and its results will be returned.
ReverseOrderPersonalNamesDomain	Specifies the domain to use when parsing reverse order personal names. The valid values are the domain names defined in the Open Parser Domain Editor too in Enterprise Designer.
ReverseOrderPersonalNamesPriority	Specify a number between 1 and 5 that indicates the priority of the reverse order personal names domain relative to the other domains that you are

Option	Description
	using. This determines the order in which you want the parsers to run. Results will be returned for the first domain that scores higher than the number set in the shortcut threshold option. If no domain reaches that threshold, results for the domain with the highest score are returned. If multiple domains reach the threshold at the same time, priority goes to the domain that was run first (determined by the order set here) and its results will be returned.
NaturalOrderConjoinedPersonalNamesDomain	Specifies the domain to use when parsing natural order conjoined personal names. The valid values are the domain names defined in the Open Parser Domain Editor too in Enterprise Designer.
NaturalOrderConjoinedPersonalNamesPriority	Specify a number between 1 and 5 that indicates the priority of the natural order conjoined personal names domain relative to the other domains that you are using. This determines the order in which you want the parsers to run. Results will be returned for the first domain that scores higher than the number set in the shortcut threshold option. If no domain reaches that threshold, results for the domain with the highest score are returned. If multiple domains reach the threshold at the same time, priority goes to the domain that was run first (determined by the order set here) and its results will be returned.
ReverseOrderConjoinedPersonalNamesDomain	Specifies the domain to use when parsing reverse order conjoined personal names. The valid values are the domain names defined in the Open Parser Domain Editor too in Enterprise Designer.
ReverseOrderConjoinedPersonalNamesPriority	Specify a number between 1 and 5 that indicates the priority of the reverse order conjoined personal names domain relative to the other domains that you are using. This determines the order in which you want the parsers to run. Results will be returned for the first domain that scores higher than the number set in the shortcut threshold option. If no domain reaches that threshold, results for the domain with the highest score are returned. If multiple domains reach the threshold at the same time, priority goes to the domain that was run first (determined by the order set here) and its results will be returned.
BusinessNamesDomain	Specifies the domain to use when parsing business names. The valid values are the domain names defined in the Open Parser Domain Editor too in Enterprise Designer.

Option	Description
BusinessNamesPriority	Specify a number between 1 and 5 that indicates the priority of the business names domain relative to the other domains that you are using. This determines the order in which you want the parsers to run. Results will be returned for the first domain that scores higher than the number set in the shortcut threshold option. If no domain reaches that threshold, results for the domain with the highest score are returned. If multiple domains reach the threshold at the same time, priority goes to the domain that was run first (determined by the order set here) and its results will be returned.

Output

Table 229: Open Name Parser Output

columnName	Format	Description
AccountDescription	String	An account description that is part of the name. For example, in "Mary Jones Account # 12345", the account description is "Account#12345".
Names	String	A hierarchical field that contains a list of parsed elements. This field is returned when you check the **Output results as list** box under Parsing Options.
Fields Related to Names of Companies		
FirmConjunction	String	Indicates that the name of a firm contains a conjunction such as "d/b/a" (doing business as), "o/a" (operating as), and "t/a" (trading as).
FirmName	String	The name of a company. For example, "Pitney Bowes".
FirmSuffix	String	The corporate suffix. For example, "Co." and "Inc."
IsFirm	String	Indicates that the name is a firm rather than an individual.
Fields Related to Names of Individual People		
Conjunction	String	Indicates that the name contains a conjunction such as "and", "or", or "&".
CultureCode	String	The culture codes contained in the input data.
CultureCodeUsedToParse	String	Identifies the culture-specific grammar that was used to parse the data. **Null (empty)** Global culture (default). **de** German. **es** Spanish. **ja** Japanese.

columnName	Format	Description
		Note: If you added your own domain using the Open Parser Domain Editor, the cultures and culture codes for that domain will appear in this field as well.
FirstName	String	The first name of a person.
GeneralSuffix	String	A person's general/professional suffix. For example, MD or PhD.
IsParsed	String	Indicates whether an output record was parsed. Values are true or false.
IsPersonal	String	Indicates whether the name is an individual rather than a firm. Values are true or false.
IsReverseOrder	String	Indicates whether the input name is in reverse order. Values are true or false.
LastName	String	The last name of a person. Includes the paternal last name.
LeadingData	String	Non-name information that appears before a name.
MaturitySuffix	String	A person's maturity/generational suffix. For example, Jr. or Sr.
MiddleName	String	The middle name of a person.
Name.	String	The personal or firm name that was provided in the input.
NameScore	String	Indicates the average score of known and unknown tokens for each name. The value of NameScore will be between 0 and 100, as defined in the parsing grammar. 0 is returned when no matches are returned.
SecondaryLastName	String	In Spanish parsing grammar, the surname of a person's mother.
TitleOfRespect	String	Information that appears before a name, such as "Mr.", "Mrs.", or "Dr."
TrailingData	String	Non-name information that appears after a name.
Fields Related to Conjoined Names		
Conjunction2	String	Indicates that a second, conjoined name contains a conjunction such as "and", "or", or "&".
Conjunction3	String	Indicates that a third, conjoined name contains a conjunction such as "and", "or", or "&".
FirmName2	String	The name of a second, conjoined company. For example, Baltimore Gas & Electric dba Constellation Energy.
FirmSuffix2	String	The suffix of a second, conjoined company.
FirstName2	String	The first name of a second, conjoined name.
FirstName3	String	The first name of a third, conjoined name.
GeneralSuffix2	String	The general/professional suffix for a second, conjoined name. For example, MD or PhD.
GeneralSuffix3	String	The general/professional suffix for a third, conjoined name. For example, MD or PhD.

columnName	Format	Description
IsConjoined	String	Indicates that the input name is conjoined. An example of a conjoined name is "John and Jane Smith".
LastName2	String	The last name of a second, conjoined name.
LastName3	String	The last name of a third, conjoined name.
MaturitySuffix2	String	The maturity/generational suffix for a second, conjoined name. For example, Jr. or Sr.
MaturitySuffix3	String	The maturity/generational suffix for a third, conjoined name. For example, Jr. or Sr.
MiddleName2	String	The middle name of a second, conjoined name.
MiddleName3	String	The middle name of a third, conjoined name.
TitleOfRespect2	String	Information that appears before a second, conjoined name, such as "Mr.", "Mrs.", or "Dr."
TitleOfRespect3	String	Information that appears before a third, conjoined name, such as "Mr.", "Mrs.", or "Dr."

9

About Spectrum Technology Platform

In this section:

- What Is Spectrum™ Technology Platform? 582
- Enterprise Data Management Architecture 583
- Spectrum™ Technology Platform Architecture 586
- Modules and Components 589

What Is Spectrum™ Technology Platform?

Spectrum™ Technology Platform is a system that improves the completeness, validity, consistency, timeliness, and accuracy of your data through data standardization, verification and enhancement. Ensuring that your data is accurate, complete, and up to date enables your firm to better understand and connect with your customers.

Spectrum™ Technology Platform aids in the design and implementation of business rules for data quality by performing the following functions.

Parsing, Name Standardization, and Name Validation

To perform the most accurate standardization you may need to break up strings of data into multiple fields. Spectrum™ Technology Platform provides advanced parsing features that enable you to parse personal names, company names, and many other terms and abbreviations. In addition, you can create your own list of custom terms to use as the basis of scan/extract operations. The Universal Name Module provides this functionality.

Deduplication and Consolidation

Identifying unique entities enables you to consolidate records, eliminate duplicates and develop "best-of-breed" records. A "best-of-breed" record is a composite record that is built using data from other records. The Advanced Matching Module and Data Normalization Module provide this functionality.

Address Validation

Address validation applies rules from the appropriate postal authority to put an address into a standard form and even validate that the address is a deliverable address. Address validation can help you qualify for postal discounts and can improve the deliverability of your mail. The Universal Addressing Module and the Address Now Module provide this functionality.

Geocoding

Geocoding is the process of taking an address and determining its geographic coordinates (latitude and longitude). Geocoding can be used for map generation, but that is only one application. The underlying location data can help drive business decisions. Reversing the process, you can enter a geocode (a point represented by a latitude and longitude coordinate) and receive address information about the geocode. The Enterprise Geocoding Module provides this functionality.

Location Intelligence

Location intelligence creates new information about your data by assessing, evaluating, analyzing and modeling geographic relationships. Using location intelligence processing you can verify locations and transform information into valuable business intelligence. The Location Intelligence Module provides this functionality.

Master Data Management

Master data management enables you to create relationship-centric master data views of your critical data assets. The Data Hub Module helps you identify influencers and non-obvious relationships, detect fraud, and improve the quality, integration, and accessibility of your information.

Tax Jurisdiction Assignment

Tax jurisdiction assignment takes an address and determines the tax jurisdictions that apply to the address's location. Assigning the most accurate tax jurisdictions can reduce financial risk and regulatory liability.

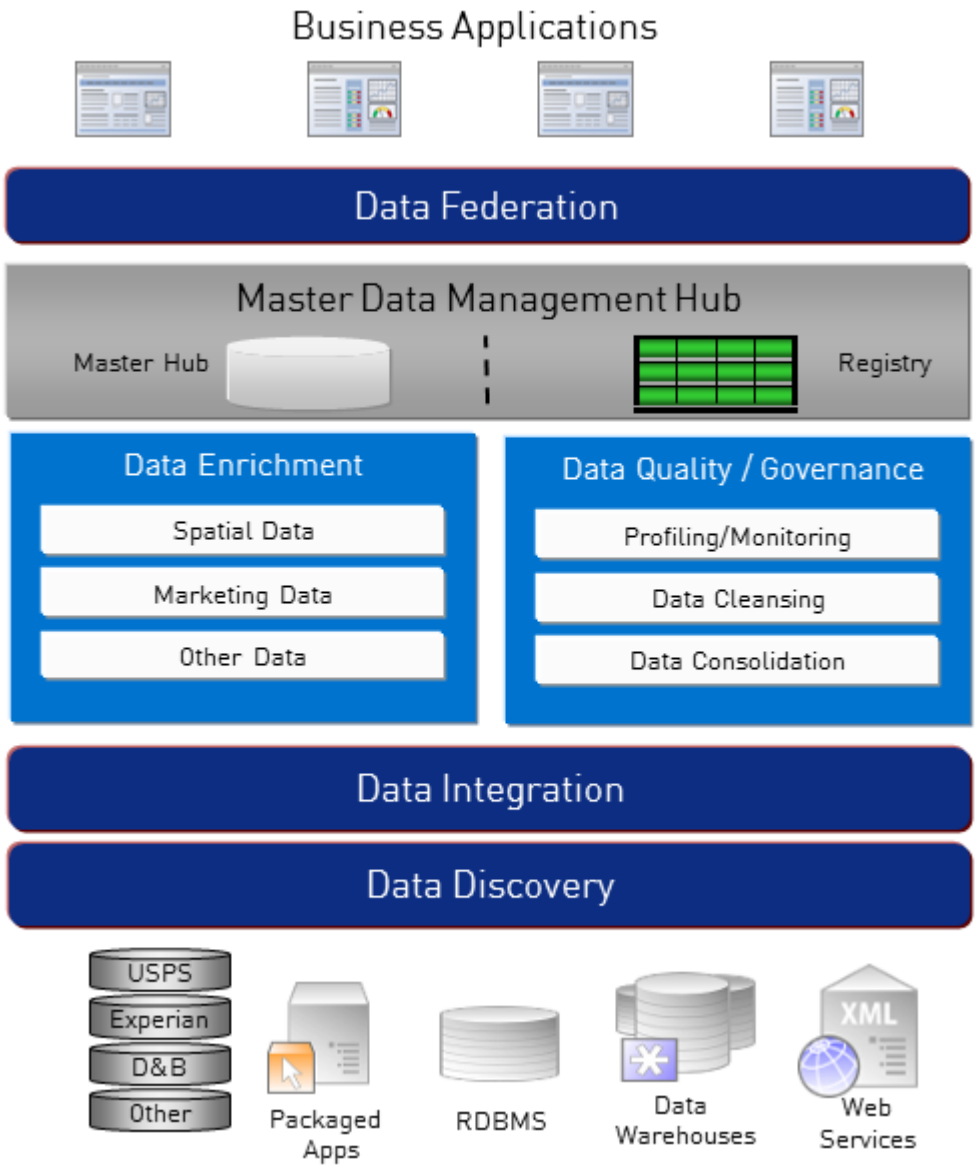
Spectrum™ Technology Platform software from Pitney Bowes Software integrates up-to-date jurisdictional boundaries with the exact street addresses of your customer records, enabling you to append the correct state, county, township, municipal, and special tax district information to your records. Some example uses of tax jurisdiction assignment are:

- Sales and use tax
- Personal property tax
- Insurance premium tax

The Enterprise Tax Module provides this functionality.

Enterprise Data Management Architecture

With Spectrum™ Technology Platform, you can build a comprehensive enterprise data management process, or you can target those individual areas in which your company needs improvement. The following diagram illustrates a complete solution that takes data from its source, through data enrichment and data quality processes, feeding a master data management hub which makes a single view of the data available to multiple business applications.



Data Discovery

Data discovery is the process of scanning your data resources to get a complete inventory of your data landscape. Spectrum™ Technology Platform can scan structured data, unstructured data, and semi-structured data using a wide array of data profiling techniques. The results of the scan are used to automatically generate a library of documentation describing your company's data assets and to create a metadata repository. This documentation and accompanying metadata repository provide the insight you need before beginning data integration, data quality, data governance, or master data management projects.

For more information on the Spectrum™ Technology Platform Data Discovery Module, contact your account executive.

Data Integration

Once you have an inventory of your data landscape, you need to consider how you will access the data you need to manage. Spectrum™ Technology Platform can connect to data in multiple sources either directly or through integration with your existing data access technologies. It supports batch and real time data integration capabilities for a variety of business needs including data warehousing, data quality,

systems integration, and migration. Spectrum™ Technology Platform can access data in RDBMS databases, data warehouses, XML files, flat files, and variable format files. Spectrum™ Technology Platform supports SQL queries with complex joins and aggregations and provides a visual query development tool. In addition, Spectrum™ Technology Platform can access data over REST and SOAP web services.

Spectrum™ Technology Platform can trigger batch processing based on the appearance of one or more source files in a specified folder. This "hot folder" trigger is useful for monitoring FTP uploads and processing them as they occur.

Some of these data integration capabilities require a license for the Enterprise Data Integration Module. For more information, contact your account executive.

Finally, Spectrum™ Technology Platform can integrate with packaged applications such as SAP and Siebel.

Data Quality/Governance

Data quality and data governance processes check your data for duplicate records, inconsistent information, and inaccurate information.

Duplicate matching identifies potential duplicate records or relationships between records, whether the data is name and address in nature or any other type of customer information. Spectrum™ Technology Platform allows you to specify a consistent set of business match rules using boolean matching methods, scoring methods, thresholds, algorithms and weights to determine if a group of records contains duplicates. Spectrum™ Technology Platform supports extensive customization so you can tailor the rules to the unique needs of your business.

Once duplicate records have been identified, you may wish to consolidate records. Spectrum™ Technology Platform allows you to specify how to link or merge duplicate records so you can create the most accurate and complete record from any collection of customer information. For example, a single best-of-breed record can be built from all of the records in a household. The Advanced Matching Module is used to identify duplicates and eliminate them.

Data quality processes also standardize your data. Standardization is a critical process because standardized data elements are necessary to achieve the highest possible results for matching and identifying relationships between records. While several modules perform standardization of one type or another, the Spectrum™ Technology Platform Data Normalization module provides the most comprehensive set of standardization features. In addition, the Universal Name module provides specific data quality features for handling personal name and business name data.

Standardized data is not necessarily accurate data. Spectrum™ Technology Platform can compare your data to known, up-to-date reference data for correctness. The sources used for this process may include regulatory bodies such as the U.S. Postal Service, third-party data providers such as Experian or D&B, or your company's internal reference sources, such as accounting data. Spectrum™ Technology Platform is particularly strong in address data validation. It can validate or standardize addresses in 250 countries and territories around the world. There are two modules that perform address validation: the Address Now Module and the Universal Addressing Module.

To determine which one is right for you, discuss your needs with your account executive.

While Spectrum™ Technology Platform can automatically handle a wide range of data quality issues, there are some situations where a manual review by a data steward is appropriate. To support this, the Business Steward Module provides a way to specify the rules that will trigger a manual review, and it provides a web-based tool for reviewing exception records. It includes integrated access to third-party tools such as Bing maps and Experian data to aid data stewards in the review and resolution process.

Data Enrichment

Data enrichment processes augment your data with additional information. Enrichment can be based on spatial data, marketing data, or data from other sources that you wish to use to add additional detail to your data. For example, if you have a database of customer addresses, you could geocode the address to determine the latitude/longitude coordinates of the address and store those coordinates as part of the

record. Your customer data could then be used to perform a variety of spatial calculations, such as finding the bank branch nearest the customer. Spectrum™ Technology Platform allows you to enrich your data with a variety of information, including geocoding (with the Enterprise Geocoding Module), tax jurisdiction assignment (with the Enterprise Tax Module), geospatial calculations (with the Location Intelligence Module), and driving and walking directions between points (with the Enterprise Routing Module).

Master Data Management Hub

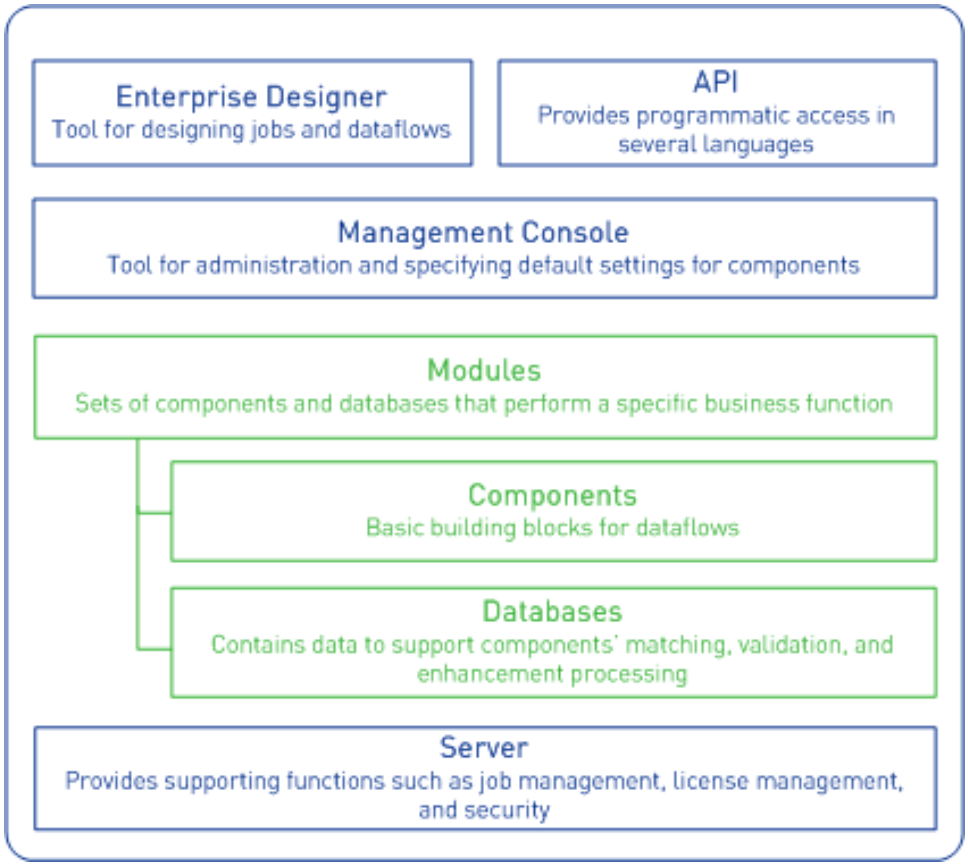
The Master Data Management (MDM) hub allows for rapid modeling of entities and their complex relationships across roles, processes and interactions. It provides built-in social network analysis capabilities to help you understand influencers, predict churn, detect non-obvious relationships and fraudulent patterns, and provide recommendations.

Spectrum™ Technology Platform supports two approaches to the MDM hub. In the master hub approach, the data is maintained in a single MDM database and applications access the data from the MDM database. In the registry approach, the data is maintained in each business application and the MDM hub registry contains keys which are used to find related records. For example, a customer's record may exist in an order entry database and a customer support database. The MDM registry would contain a single key which could be used to access the customer data in both places.

The Data Hub Module provides MDM capabilities.

Spectrum™ Technology Platform Architecture

Spectrum™ Technology Platform software from Pitney Bowes Software includes a server that supports a number of modules. These modules provide different functions, such as address validation, geocoding, and advanced parsing, among others. The following diagram illustrates the Spectrum™ Technology Platform architecture.


Enterprise Designer
Tool for designing jobs and dataflows
API
Provides programmatic access in several languages
Management Console
Tool for administration and specifying default settings for components
Modules
Sets of components and databases that perform a specific business function
Components
Basic building blocks for dataflows
Databases
Contains data to support components' matching, validation, and enhancement processing
Server
Provides supporting functions such as job management, license management, and security

Server

The foundation of the Spectrum™ Technology Platform is the server. The server handles data processing, synchronizes repository data, and manages communication between the client and the transformation modules via TCP/IP. It provides job management and security features.

Modules

Modules are sets of features that perform a specific function. For example, if you want to standardize your customers' addresses to conform to USPS standards, you would license the Universal Addressing module. If you want to determine the tax jurisdictions that apply to each of your customers, you would license the Enterprise Tax module. You can license just one module or multiple modules, depending on your specific needs. Most modules consist of "components" and databases.

Components

A component is a basic building block in a customer data quality process. Each component performs a specific function. For example, the Enterprise Geocoding module's GeocodeUSAddress component takes an address and returns the latitude and longitude coordinates for that address; the Universal Addressing module's GetCityStateProvince takes a postal code and returns the city and state/province where that postal code is located.

Some components must first be combined with other components into a job, service, or subflow before they can be executed. Use Enterprise Designer to create jobs, services, subflows, and process flows. For more information, see **Enterprise Designer** on page 588.

The components that you have available on your system depend on which Spectrum™ Technology Platform modules you have licensed from Pitney Bowes Software.

Databases

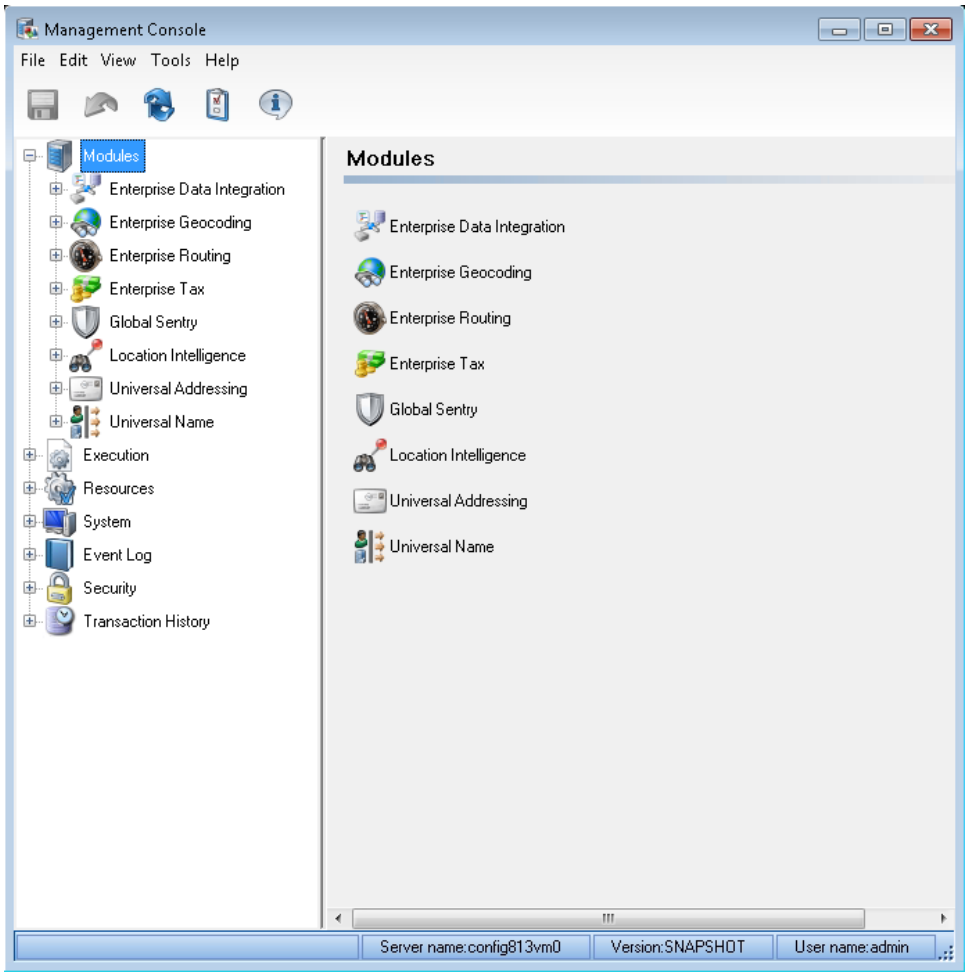
Modules often include databases that contain the data needed by the components in the module. For example, the Universal Addressing module needs to have access to USPS data in order to verify and standardize addresses. So, the Universal Addressing module comes with the U.S. Postal Database, which you must load into a location that is accessible by your Spectrum™ Technology Platform system.

Modules have both required and optional databases. Optional databases provide data needed for certain features that can greatly enhance your Spectrum™ Technology Platform process.

Management Console

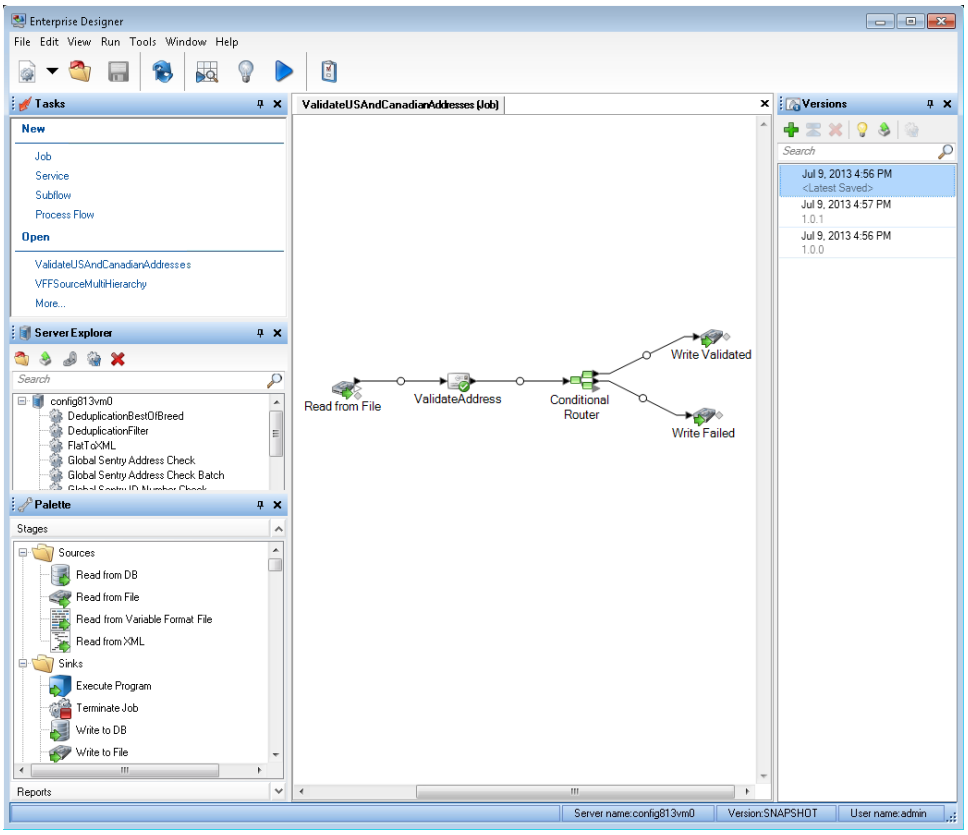
The Management Console is a Windows-based tool for administering Spectrum™ Technology Platform. You can use the Management Console to:

- Specify a server access address
- Select component access method (local or hosted)
- Specify the default settings for Spectrum™ Technology Platform components
- Manage user accounts, including permissions and passwords
- Set up logging, tracking, and reporting.



Enterprise Designer

Enterprise Designer is a Windows-based tool for creating Spectrum™ Technology Platform jobs, services, subflows, and process flows. It utilizes an easy drag-and-drop interface to allow you to graphically create complex dataflows.



API

The Spectrum™ Technology Platform API, which is provided in the Spectrum™ Technology Platform Client API, is designed to provide simple integration, streamline record processing, and support backward compatibility of future versions. The Spectrum™ Technology Platform API can be accessed through:

- C
- C++
- COM
- Java
- .NET
- Web services

Modules and Components

Table 230: Modules and Components

Module	Description	Components
Address Now Module	Provides enhanced validation and standardization for addresses outside the U.S., and other address processing.	Build Global Address Get Global Candidate Addresses Validate Global Address
Advanced Matching Module	Matches records within and/or between input files.	Best Of Breed

Module	Description	Components
		Candidate Finder Duplicate Synchronization Filter Interflow Match Intraflow Match Match Key Generator Transactional Match
Business Steward Module	Identifies exception records and provides a browser-based tool for manually reviewing exception records.	Exception Monitor Read Exceptions Write Exceptions
Country Identifier	Takes a country name or a combination of postal code and state/province and returns the two-character ISO country code, the three-character Universal Postal Union (UPU) code, and the English country name.	Country Identifier
Data Hub Module	Links and analyzes data, identifying relationships and trends.	Write to Hub Read From Hub Query Hub Graph Visualization
Data Integration Module	Provides capabilities useful in data warehousing, data quality, systems integration, and migration.	Field Selector Generate Time Dimension Query Cache Write to Cache
Data Normalization Module	Removes inconsistencies in data.	Advanced Transformer Open Parser Table Lookup Transliterator
Enterprise Data Integration	Connects to data in multiple sources for a variety of business needs including data warehousing, data quality, systems integration, and migration.	Call Stored Procedure Field Selector Generate Time Dimension Query Cache Write to Cache
Enterprise Geocoding Module	Determines the geographic coordinates for an address. Also determines the address of a given latitude and longitude.	GeocodeAddressAUS GeocodeAddressGBR GeocodeAddressGlobal GeocodeAddressWorld GeocodeUSAddress GNAFPIDLocationSearch ReverseAPNLookup ReverseGeocodeAddressGlobal

Module	Description	Components
		ReverseGeocodeUSLocation
Enterprise Routing Module	Obtains driving or walking directions, calculates drive time and drive distance, and identifies locations within a certain time or distance from a starting point.	GetRouteData GetTravelBoundary GetTravelCostMatrix GetTravelDirections PersistentUpdate
Enterprise Tax Module	Determines the tax jurisdictions that apply to a given location.	AssignGeoTAXInfo CalculateDistance
GeoConfidence Module	Determines the probability that an address or street intersection is within a given area.	Geo Confidence Surface CreatePointsConvexHull
Global Sentry	Attempts to match transactions against government-provided watch lists that contain data from different countries.	Global Sentry Global Sentry Address Check Global Sentry ID Number Check Global Sentry Name Check Global Sentry Other Data Check
Location Intelligence Module	Performs point in polygon and radial analysis against a variety of geospatial databases.	Closest Site Find Nearest Point In Polygon Query Spatial Data Read Spatial Data Spatial Calculator Spatial Union
SAP Module	Enables Spectrum™ Technology Platform to interface with SAP Customer Relationship Management Module applications.	SAP Generate Match Key SAP Generate Match Score SAP Generate Search Key SAP Generate Search Key Constant SAP Generate Search Key Metaphone SAP Generate Search Key Substring SAP Validate Address With Candidates
Siebel Module	Enables Spectrum™ Technology Platform to interface Siebel applications.	Siebel Generate Match Key Siebel Generate Match Score Siebel Generate Search Key Siebel Business Name Standardization Siebel Standardize Name

Module	Description	Components
		Siebel Geocode US Address With Candidates
		Siebel Geocode US Address With No Candidates
		Siebel Get Global Candidate Addresses
		Siebel Validate Address With Candidates
		Siebel Validate Address With No Candidates
Universal Addressing Module	Standardizes and validates addresses according to the postal authority's standards.	Get Candidate Addresses
		Get City State Province
		Get Postal Codes
		Validate Address
		Validate Address AUS
		Validate Address Global
Universal Name Module	Parses personal names, company names, addresses, and many other terms and abbreviations.	Name Parser (Deprecated)
		Name Variant Finder
		Open Name Parser

Appendix

In this section:

- Country ISO Codes and Module Support 595

A

Country ISO Codes and Module Support

In this section:

- Country ISO Codes and Module Support 596

Country ISO Codes and Module Support

The following table lists the ISO codes for each country as well as the modules that support addressing, geocoding, and routing for each country.

Note that the Enterprise Geocoding Module includes databases for Africa (30 countries), Middle East (8 countries) and Latin America (20 countries). These databases cover the smaller countries in those regions that do not have their own country-specific geocoding databases. The Supported Modules column indicates which countries are covered by these Africa, Middle East, and Latin America databases.

Also, the Geocode Address World database provides geographic and limited postal geocoding (but not street-level geocoding) for all countries.

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Afghanistan	AF	AFG	Address Now Module Universal Addressing Module
Aland Islands	AX	ALA	Address Now Module Universal Addressing Module
Albania	AL	ALB	Address Now Module Universal Addressing Module
Algeria	DZ	DZA	Address Now Module Universal Addressing Module
American Samoa	AS	ASM	Address Now Module Universal Addressing Module
Andorra	AD	AND	Address Now Module Enterprise Geocoding Module[12] Universal Addressing Module GeoComplete Module
Angola	AO	AGO	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Anguilla	AI	AIA	Address Now Module Universal Addressing Module
Antarctica	AQ	ATA	Address Now Module Universal Addressing Module
Antigua And Barbuda	AG	ATG	Address Now Module Universal Addressing Module
Argentina	AR	ARG	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Armenia	AM	ARM	Address Now Module Universal Addressing Module

[12] Andorra is covered by the Spain geocoder

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Aruba	AW	ABW	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Australia	AU	AUS	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Austria	AT	AUT	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Azerbaijan	AZ	AZE	Address Now Module Universal Addressing Module
Bahamas	BS	BHS	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Bahrain	BH	BHR	Address Now Module Enterprise Geocoding Module (Middle East) Enterprise Routing Module Universal Addressing Module
Bangladesh	BD	BGD	Address Now Module Universal Addressing Module
Barbados	BB	BRB	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Belarus	BY	BLR	Address Now Module Universal Addressing Module
Belgium	BE	BEL	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Belize	BZ	BLZ	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Benin	BJ	BEN	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Bermuda	BM	BMU	Address Now Module Universal Addressing Module
Bhutan	BT	BTN	Address Now Module Universal Addressing Module
Bolivia, Plurinational State Of	BO	BOL	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Bonaire, Saint Eustatius And Saba	BQ	BES	Address Now Module Universal Addressing Module
Bosnia And Herzegovina	BA	BIH	Address Now Module Universal Addressing Module
Botswana	BW	BWA	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Bouvet Island	BV	BVT	Address Now Module Universal Addressing Module
Brazil	BR	BRA	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
British Indian Ocean Territory	IO	IOT	Address Now Module Universal Addressing Module
Brunei Darussalam	BN	BRN	Address Now Module Universal Addressing Module
Bulgaria	BG	BGR	Address Now Module Universal Addressing Module
Burkina Faso	BF	BFA	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Burundi	BI	BDI	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Cambodia	KH	KHM	Address Now Module Universal Addressing Module
Cameroon	CM	CMR	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Canada	CA	CAN	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Cape Verde	CV	CPV	Address Now Module Universal Addressing Module
Cayman Islands	KY	CYM	Address Now Module Universal Addressing Module
Central African Republic	CF	CAF	Address Now Module Universal Addressing Module
Chad	TD	TCD	Address Now Module Universal Addressing Module
Chile	CL	CHL	Address Now Module Enterprise Geocoding Module Universal Addressing Module GeoComplete Module
China	CN	CHN	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Christmas Island	CX	CXR	Address Now Module Universal Addressing Module
Cocos (Keeling) Islands	CC	CCK	Address Now Module Universal Addressing Module
Colombia	CO	COL	Address Now Module Universal Addressing Module
Comoros	KM	COM	Address Now Module Universal Addressing Module
Congo	CG	COG	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Congo, The Democratic Republic Of The	CD	COD	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Cook Islands	CK	COK	Address Now Module Universal Addressing Module
Costa Rica	CR	CRI	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Côte d'Ivoire	CI	CIV	Address Now Module Universal Addressing Module
Croatia	HR	HRV	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Cuba	CU	CUB	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Curacao	CW	CUW	Address Now Module Universal Addressing Module
Cyprus	CY	CYP	Address Now Module Universal Addressing Module
Czech Republic	CZ	CZE	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Denmark	DK	DNK	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Djibouti	DJ	DJI	Address Now Module Universal Addressing Module
Dominica	DM	DMA	Address Now Module Universal Addressing Module
Dominican Republic	DO	DOM	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Ecuador	EC	ECU	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Egypt	EG	EGY	Address Now Module Enterprise Geocoding Module (Middle East) Enterprise Routing Module Universal Addressing Module
El Salvador	SV	SLV	Address Now Module Enterprise Geocoding Module (Latin

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
			America) Universal Addressing Module
Equatorial Guinea	GQ	GNQ	Address Now Module Universal Addressing Module
Eritrea	ER	ERI	Address Now Module Universal Addressing Module
Estonia	EE	EST	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Ethiopia	ET	ETH	Address Now Module Universal Addressing Module
Falkland Islands (Malvinas)	FK	FLK	Address Now Module Universal Addressing Module
Faroe Islands	FO	FRO	Address Now Module Universal Addressing Module
Fiji	FJ	FJI	Address Now Module Universal Addressing Module
Finland	FI	FIN	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
France	FR	FRA	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
French Guiana	GF	GUF	Address Now Module Enterprise Geocoding Module [13] Universal Addressing Module
French Polynesia	PF	PYF	Address Now Module Universal Addressing Module
French Southern Territories	TF	ATF	Address Now Module Universal Addressing Module
Gabon	GA	GAB	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module

[13] French Guiana is covered by the France geocoder

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Gambia	GM	GMB	Address Now Module Universal Addressing Module
Georgia	GE	GEO	Address Now Module Universal Addressing Module
Germany	DE	DEU	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Ghana	GH	GHA	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Gibraltar	GI	GIB	Address Now Module Enterprise Geocoding Module [14]Universal Addressing Module
Greece	GR	GRC	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Greenland	GL	GRL	Address Now Module Universal Addressing Module
Grenada	GD	GRD	Address Now Module Universal Addressing Module
Guadeloupe	GP	GLP	Address Now Module Enterprise Geocoding Module [15] Universal Addressing Module
Guam	GU	GUM	Address Now Module Universal Addressing Module
Guatemala	GT	GTM	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Guernsey	GG	GGY	Address Now Module Universal Addressing Module
Guinea	GN	GIN	Address Now Module Universal Addressing Module
Guinea-Bissau	GW	GNB	Address Now Module Universal Addressing Module

[14] Gibraltar is covered by the Spain geocoder
[15] Guadeloupe is covered by the France geocode

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Guyana	GY	GUY	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Haiti	HT	HTI	Address Now Module Universal Addressing Module
Heard Island and McDonald Islands	HM	HMD	Address Now Module Universal Addressing Module
Holy See (Vatican City State)	VA	VAT	Address Now Module Enterprise Geocoding Module [16] Universal Addressing Module
Honduras	HN	HND	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Hong Kong	HK	HKG	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Hungary	HU	HUN	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Iceland	IS	ISL	Address Now Module Universal Addressing Module
India	IN	IND	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Indonesia	ID	IDN	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Iran, Islamic Republic Of	IR	IRN	Address Now Module Universal Addressing Module
Iraq	IQ	IRQ	Address Now Module Universal Addressing Module
Ireland	IE	IRL	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module

[16] The Vatican is covered by the Italy geocoder

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Isle Of Man	IM	IMN	Address Now Module Universal Addressing Module
Israel	IL	ISR	Address Now Module Universal Addressing Module
Italy	IT	ITA	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Jamaica	JM	JAM	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Japan	JP	JPN	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Jersey	JE	JEY	Address Now Module Universal Addressing Module
Jordan	JO	JOR	Address Now Module Universal Addressing Module
Kazakhstan	KZ	KAZ	Address Now Module Universal Addressing Module
Kenya	KE	KEN	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Kiribati	KI	KIR	Address Now Module Universal Addressing Module
Korea, Democratic People's Republic Of	KP	PRK	Address Now Module Universal Addressing Module
Korea, Republic Of	KR	KOR	Address Now Module Universal Addressing Module
Kosovo	KS	KOS	Address Now Module Universal Addressing Module GeoComplete Module
Kuwait	KW	KWT	Address Now Module Enterprise Geocoding Module (Middle East) Enterprise Routing Module Universal Addressing Module

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Kyrgyzstan	KG	KGZ	Address Now Module Universal Addressing Module
Lao People's Democratic Republic	LA	LAO	Address Now Module Universal Addressing Module
Latvia	LV	LVA	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Lebanon	LB	LBN	Address Now Module Enterprise Geocoding Module (Middle East) Universal Addressing Module
Lesotho	LS	LSO	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Liberia	LR	LBR	Address Now Module Universal Addressing Module
Libyan Arab Jamahiriya	LY	LBY	Address Now Module Universal Addressing Module
Liechtenstein	LI	LIE	Address Now Module Enterprise Geocoding Module [17] Enterprise Routing Module Universal Addressing Module GeoComplete Module
Lithuania	LT	LTU	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Luxembourg	LU	LUX	Address Now Module Enterprise Geocoding Module [18] Enterprise Routing Module Universal Addressing Module GeoComplete Module
Macao	MO	MAC	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Macedonia, Former Yugoslav Republic Of	MK	MKD	Address Now Module Universal Addressing Module
Madagascar	MG	MDG	Address Now Module Universal Addressing Module

[17] Liechtenstein is covered by the Switzerland geocoder
[18] Luxembourg is covered by the Belgium geocoder

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Malawi	MW	MWI	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Malaysia	MY	MYS	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Maldives	MV	MDV	Address Now Module Universal Addressing Module
Mali	ML	MLI	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Malta	ML	MLT	Enterprise Routing Module Address Now Module Universal Addressing Module
Marshall Islands	MH	MHL	Address Now Module Universal Addressing Module
Martinique	MQ	MTQ	Address Now Module Enterprise Geocoding Module Guadeloupe is covered by the France geocode Universal Addressing Module
Mauritania	MR	MRT	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Mauritius	MU	MUS	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Mayotte	YT	MYT	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Mexico	MX	MEX	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Micronesia, Federated States Of	FM	FSM	Address Now Module Universal Addressing Module
Moldova, Republic Of	MD	MDA	Address Now Module Universal Addressing Module

[19] Martinique is covered by the France geocoder.
[20] Mayotte is covered by the France geocoder.

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Monaco	MC	MCO	Address Now Module Enterprise Geocoding Module [21]Universal Addressing Module
Mongolia	MN	MNG	Address Now Module Universal Addressing Module
Montenegro	ME	MNE	Address Now Module Universal Addressing Module
Montserrat	MS	MSR	Address Now Module Universal Addressing Module
Morocco	MA	MAR	Address Now Module Enterprise Geocoding Module (Africa) Enterprise Routing Module Universal Addressing Module
Mozambique	MZ	MOZ	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Myanmar	MM	MMR	Address Now Module Universal Addressing Module
Namibia	NA	NAM	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Nauru	NR	NRU	Address Now Module Universal Addressing Module
Nepal	NP	NPL	Address Now Module Universal Addressing Module
Netherlands	NL	NLD	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
New Caledonia	NC	NCL	Address Now Module Universal Addressing Module
New Zealand	NZ	NZL	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Nicaragua	NI	NIC	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module

[21] Monaco is covered by the France geocoder

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Niger	NE	NER	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Nigeria	NG	NGA	Address Now Module Enterprise Geocoding Module (Africa) Enterprise Routing Module Universal Addressing Module
Niue	NU	NIU	Address Now Module Universal Addressing Module
Norfolk Island	NF	NFK	Address Now Module Universal Addressing Module
Northern Mariana Islands	MP	MNP	Address Now Module Universal Addressing Module
Norway	NO	NOR	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Oman	OM	OMN	Address Now Module Enterprise Geocoding Module (Middle East) Enterprise Routing Module Universal Addressing Module
Pakistan	PK	PAK	Address Now Module Universal Addressing Module
Palau	PW	PLW	Address Now Module Universal Addressing Module
Palestinian Territory, Occupied	PS	PSE	Address Now Module Universal Addressing Module
Panama	PA	PAN	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Papua New Guinea	PG	PNG	Address Now Module Universal Addressing Module
Paraguay	PY	PRY	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Peru	PE	PER	Address Now Module Enterprise Geocoding Module (Latin

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
			America) Universal Addressing Module
Philippines	PH	PHL	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Pitcairn	PN	PCN	Address Now Module Universal Addressing Module
Poland	PL	POL	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Portugal	PT	PRT	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Puerto Rico	PR	PRI	Address Now Module Universal Addressing Module
Qatar	QA	QAT	Address Now Module Enterprise Geocoding Module (Middle East) Enterprise Routing Module Universal Addressing Module
Reunion	RE	REU	Address Now Module Enterprise Geocoding Module [22]Universal Addressing Module
Romania	RO	ROU	Address Now Module Universal Addressing Module Enterprise Routing Module
Russian Federation	RU	RUS	Address Now Module Enterprise Geocoding Module Universal Addressing Module GeoComplete Module
Rwanda	RW	RWA	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Saint Barthelemy	BL	BLM	Address Now Module Universal Addressing Module

[22] Reunion is covered by the France geocoder

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Saint Helena, Ascension & Tristan Da Cunha	SH	SHE	Address Now Module Universal Addressing Module
Saint Kitts and Nevis	KN	KNA	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Saint Lucia	LC	LCA	Address Now Module Universal Addressing Module
Saint Martin (French Part)	MF	MAF	Address Now Module Universal Addressing Module
Saint Pierre and Miquelon	PM	SPM	Address Now Module Universal Addressing Module
Saint Vincent And The Grenadines	VC	VCT	Address Now Module Universal Addressing Module
Samoa	WS	WSM	Address Now Module Universal Addressing Module
San Marino	SM	SMR	Address Now Module Enterprise Geocoding Module [23] Universal Addressing Module
Sao Tome And Principe	ST	STP	Address Now Module Universal Addressing Module
Saudi Arabia	SA	SAU	Address Now Module Enterprise Geocoding Module (Middle East) Universal Addressing Module
Senegal	SN	SEN	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Serbia	RS	SRB	Address Now Module Universal Addressing Module
Seychelles	SC	SYC	Address Now Module Universal Addressing Module
Sierra Leone	SL	SLE	Address Now Module Universal Addressing Module
Singapore	SG	SGP	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module

[23] San Marino is covered by the Italy geocoder

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Sint Maarten (Dutch Part)	SX	SXM	Universal Addressing Module
Slovakia	SK	SVK	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Slovenia	SI	SVN	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Solomon Islands	SB	SLB	Address Now Module Universal Addressing Module
Somalia	SO	SOM	Address Now Module Universal Addressing Module
South Africa	ZA	ZAF	Address Now Module Enterprise Geocoding Module Universal Addressing Module GeoComplete Module
South Georgia And The South Sandwich Islands	GS	SGS	Address Now Module Enterprise Geocoding Module Universal Addressing Module
South Sudan	SS	SSD	Address Now Module Universal Addressing Module
Spain	ES	ESP	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Sri Lanka	LK	LKA	Address Now Module Universal Addressing Module
Sudan	SD	SDN	Address Now Module Universal Addressing Module
Suriname	SR	SUR	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Svalbard And Jan Mayen	SJ	SJM	Address Now Module Universal Addressing Module
Swaziland	SZ	SWZ	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Sweden	SE	SWE	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Switzerland	CH	CHE	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
Syrian Arab Republic	SY	SYR	Address Now Module Universal Addressing Module
Taiwan, Province of China	TW	TWN	Enterprise Routing Module Address Now Module Universal Addressing Module
Tajikistan	TJ	TJK	Address Now Module Universal Addressing Module
Tanzania, United Republic Of	TZ	TZA	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Thailand	TH	THA	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
Timor-Leste	TL	TLS	Address Now Module Universal Addressing Module
Togo	TG	TGO	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Tokelau	TK	TKL	Address Now Module Universal Addressing Module
Tonga	TO	TON	Address Now Module Universal Addressing Module
Trinidad and Tobago	TT	TTO	Address Now Module Enterprise Geocoding Module (Latin America) Universal Addressing Module
Tunisia	TN	TUN	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Turkey	TR	TUR	Address Now Module Enterprise Geocoding Module

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
			Universal Addressing Module GeoComplete Module
Turkmenistan	TM	TKM	Address Now Module Universal Addressing Module
Turks And Caicos Islands	TC	TCA	Address Now Module Universal Addressing Module
Tuvalu	TV	TUV	Address Now Module Universal Addressing Module
Uganda	UG	UGA	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Ukraine	UA	UKR	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module
United Arab Emirates	AE	ARE	Address Now Module Enterprise Geocoding Module (Middle East) Enterprise Routing Module Universal Addressing Module
United Kingdom	GB	GBR	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
United States	US	USA	Address Now Module Enterprise Geocoding Module Enterprise Routing Module Universal Addressing Module GeoComplete Module
United States Minor Outlying Islands	UM	UMI	Address Now Module Universal Addressing Module
Uruguay	UY	URY	Address Now Module Enterprise Geocoding Module Universal Addressing Module
Uzbekistan	UZ	UZB	Address Now Module Universal Addressing Module
Vanuatu	VU	VUT	Address Now Module Universal Addressing Module
Venezuela, Bolivarian Republic Of	VE	VEN	Address Now Module Enterprise Geocoding Module Universal Addressing Module

ISO Country Name	ISO 3116-1 Alpha-2	ISO 3116-1 Alpha-3	Supported Modules
Viet Nam	VN	VNM	Enterprise Routing Module Address Now Module Universal Addressing Module
Virgin Islands, British	VG	VGB	Address Now Module Universal Addressing Module
Virgin Islands, U.S.	VI	VIR	Address Now Module Universal Addressing Module
Wallis and Futuna	WF	WLF	Address Now Module Universal Addressing Module
Western Sahara	EH	ESH	Address Now Module Universal Addressing Module
Yemen	YE	YEM	Address Now Module Universal Addressing Module
Zambia	ZM	ZMB	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module
Zimbabwe	ZW	ZWE	Address Now Module Enterprise Geocoding Module (Africa) Universal Addressing Module

Notices

© 2014 Pitney Bowes Software Inc. All rights reserved. MapInfo and Group 1 Software are trademarks of Pitney Bowes Software Inc. All other marks and trademarks are property of their respective holders.

USPS® Notices

Pitney Bowes Inc. holds a non-exclusive license to publish and sell ZIP + 4® databases on optical and magnetic media. The following trademarks are owned by the United States Postal Service: CASS, CASS Certified, DPV, eLOT, FASTforward, First-Class Mail, Intelligent Mail, LACS^{Link}, NCOA^{Link}, PAVE, PLANET Code, Postal Service, POSTNET, Post Office, RDI, Suite^{Link} , United States Postal Service, Standard Mail, United States Post Office, USPS, ZIP Code, and ZIP + 4. This list is not exhaustive of the trademarks belonging to the Postal Service.

Pitney Bowes Inc. is a non-exclusive licensee of USPS® for NCOA^{Link}® processing.

Prices for Pitney Bowes Software's products, options, and services are not established, controlled, or approved by USPS® or United States Government. When utilizing RDI™ data to determine parcel-shipping costs, the business decision on which parcel delivery company to use is not made by the USPS® or United States Government.

Data Provider and Related Notices

Data Products contained on this media and used within Pitney Bowes Software applications are protected by various trademarks and by one or more of the following copyrights:

© Copyright United States Postal Service. All rights reserved.

© 2014 TomTom. All rights reserved. TomTom and the TomTom logo are registered trademarks of TomTom N.V.

© Copyright NAVTEQ. All rights reserved

Data © 2014 NAVTEQ North America, LLC

Fuente: INEGI (Instituto Nacional de Estadística y Geografía)

Based upon electronic data © National Land Survey Sweden.

© Copyright United States Census Bureau

© Copyright Nova Marketing Group, Inc.

Portions of this program are © Copyright 1993-2007 by Nova Marketing Group Inc. All Rights Reserved

© Copyright Second Decimal, LLC

© Copyright Canada Post Corporation

This CD-ROM contains data from a compilation in which Canada Post Corporation is the copyright owner.

© 2007 Claritas, Inc.

The Geocode Address World data set contains data licensed from the GeoNames Project (**www.geonames.org**) provided under the Creative Commons Attribution License ("Attribution License") located at **http://creativecommons.org/licenses/by/3.0/legalcode**. Your use of the GeoNames data (described in the Spectrum™ Technology Platform User Manual) is governed by the terms of the Attribution License, and any conflict between your agreement with Pitney Bowes Software, Inc. and the Attribution License will be resolved in favor of the Attribution License solely as it relates to your use of the GeoNames data.

ICU Notices

Copyright © 1995-2011 International Business Machines Corporation and others.

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the

Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.